V. JANKOVIĆ / D. TANASKOSKI / N. ČAKLOVIĆ

SPEKTRUM

priručnik








dipl. ing. Vladimir Janković
dipl. ing. Dragan Tanaskoski
dipl. ing. Nenad Čaklović

Spektrum priručnik

Četvrto izdanje



Mikro knjiga

Spektrum priručnik
YU ISBN 86-80003-01-8
UDK. 68.31.-181.48

Adresa izdevača
Mikro knjige, P.O.Box 75, 11090 Rekovica-Beograd

Prvo izdanje, 1985
Drugo izdanje, 1985
Treće izdanje, 1986
Četvrto izdanje, 1987, tiraž 4000

Foto slog "Službeni list SFRJ", Beograd
Štampa "Mileševo", Prijepolje

Beograd 1987

izdavač
Samostalno izdanje grupe autora
Vladimir Janković
Dragan Tanaskoski
Nenad Čaklović

recenzent
Ladislav Rupnik

lektor
Aleksandar Janković

korektor
Nenad Čaklović

tehnički urednik
Marina Vuga

fotografija na koricama
Dušan Milovanović

Predgovor

Ova knjiga je nastala sa namarom da se istovremeno omogući upoznavanje sa Spektrumom i mikroračunarima uopšte, kao i da se prikaže što veći deo materije koja je potrebna za njegovu potpunu i kreativnu upotrebu.

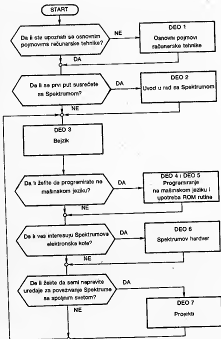
Način korišćenja knjige zavisi od prethodnog znanja čitaoca i od željene primene računara. U skladu sa tim treba izabrati delove knjige koji će biti od interesa. U tome može pomoći dijagram toka koji je prikazan na sledećoj strani.

Autori se zahvaljuju svima onima koji su pomogli da se knjiga realizuje.

Predgovor drugom izdanju

Izbor tema i način njihove obrade u ovoj knjizi pokazao se opravdanim. Velika potreba za knjigom ovakvog profila dovela je do, za naše prilike retkog događaja, da se u istoj godini pojavljuje drugo izdanja istog naslova.

U drugom izdanju autori su izvršili ispravka uočenih štamparskih grešaka.


START
Da li ste upoznati sa osnovnim pojmovima računarske tehnike?
NE
DEO 1
Osnovni pojmovi računarske tehnike
DA
Da li se prvi put susrećete sa Spektrumom?
DA
DEO 2
Uvod u rad sa Spektrumom
NE
DEO 3
Bejzik
Da li želite da programirate na mašinskom jeziku?
DA
DEO 4 i DEO 5
Programiranje na mašinskom jeziku i upotreba ROM rutine
NE
Da li vas interesuju Spektrumova elektronska kola?
DA
DEO 6
Spektrumov hardver
NE
Da li želite da sami napravite uređaje za povezivanje Spektruma sa spoljnim svetom?
DA
DEO 7
Projekti
NE

Uvod

U proleće 1980. godina mala engleska firma Science of Cambridge, do tada poznata po minijaturnim i jeftinim komponentama za HI-FI linije, izbacuje na tržište mikroračunar ZX-80. To je bio prvi jeftin i pouzdan uređaj koji je sa TV prijemnikom i magnetofonom činio kućni mikroračunarski sistem. Sledeće godine firma menja naziv u Sinclair Research, a ZX-80 zamenjuje poboljšanom varijantom ZX-81. Godinu dana kasnije na tržištu se pojavljuje ZX spektrum koji predstavlja značajan korak ka kvalitetu i novim mogućnostima. Povoljan odnos cena i onoga što se dobija objašnjava činjenicu da je početkom 1984. godine prodat i milioniti primerak.

Kućni računari čine posebnu grupu mikroračunara namenjenih prvenstveno za ličnu i, delimično, za poslovnu upotrebu. Mikroračunari po ceni, dimenzijama i karakteristikama predstavljaju znatno redukovane računarske sisteme koji se koristi u poslovnim, naučnim, industrijskim i vojnim primenama. ZX Spektrum poseduje sve osobine kompletnog računara: moćan i komforan programski jezik BASIC (bejzik) kojim se korisnik obraća računaru, veliku memoriju koja omogućuje formiranje dugih programa i skladištenje većeg broja podataka, grafiku visoke rezolucije u boji, što omogućava kvalitetno i atraktivno prikazivanje slike na ekranu, kao i generisanje tonova. Dalja nadgradnja sistema uređajima za brže unošenje i skladištenja podataka i uređajima za ispisivanje podataka ostvaruje mogućnost ozbiljnije upotrebe. Matematička izračunavanja, vođenje kućnih i manjih poslovnih finansija, obrada teksta, formiranje baza podataka, merenje i upravljanje uređajima i procesima su, uz video igra, najčešće primene.

Osnovni pojmovi o računarima

1

1-1 RAČUNAR, HARDVER, SOFTVER

Računar je mašina koja automatski obrađuje veliku količinu podataka (informacija).

Hardver (engl. hardware) računara čine njegove fizička komponente: integrisana kola, tranzistori, otpornici, kondenzatori, provodnici i drugo.

Naredba su posebne vrste podataka kojima se računaru poručuja šta treba da uradi. Niz naredbi čini program. Računar obavlja željeni zadatak izvršavajući naredbu po naredbu programa.

Programi i podaci kojima računar raspolaže, bilo da se već nalaza u njemu, bilo da se nalaze u spoljnoj sradini, čine softver (engl software) računara.

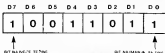
1-2 BINARNI BROJ, BIT, BAJT

U računaru su podaci predstavljeni samo brojevima koji se sastoje od niza nula i jedinica. To su, takozvani, binarni brojevi. Sva slova, znaci, decimalni i drugi brojevi se predstavljaju, odnosno, koduju pomoću binarnih brojeva i u takvom obliku ih računar pamti i obrađuja.

Fizički gledano, za računar je informacija data jednim od dva moguća stanja. Stanja označena nulom ili jedinicom mogu biti pradstavljena postojanjem ili odsustvom napona ili struje, širinom impulsa zabaleženog na magnetofonskoj traci i na druge načine.

Takva informacija, odnosno podatak, koji može imati samo jednu od dve vrednosti naziva se bit. To je i jadinica mera za količinu informacija (kao što je metar jadinica mere za dužinu). Osam bita čine bajt. On može imati jednu od 256 različitih vrednosti (2^8), jer je to broj različitih kombinacija od osam cifara koje mogu biti nula ili jedan. Veće jedinice su 1 Kbit, kilobit = 1024 bita (2^{10}), 1 Mbit, me-

gabit = 2^{20} bita, 1 Kbajt, kilobajt = 1024 bajta, 1 Mbajt, megabajt = 2^{20} bajta.



Slika 1-1 Simbolički prikaz jednog bajta

1-3 MIKROPROCESOR, MEMORIJA, PERIFERNE JEDINICE

Računar se sastoji od kola koja razlikuju samo dve različite vrednosti napona, jedna odgovara nuli, a druga jedinici. Takva kola se nazivaju logičkim, odnosno, digitalnim kolima. Svako od njih se sastoji od desetak pa do nakoliko hiljada tranzistora zapakovanih u plastično ili keramičko kućište. Na taj način se dobija integrisano kolo, odnosno, čip.

Mikroprocesor rukovodi radom računara. To je tzv. centralna procasorska jedinica upakovana u samo jedno integrisano kolo. On redom iz memorije čita i izvršava naredbu po naredbu. Sastoji se iz komandnog organa, izvršnog organa i malo memorije. Spektrumov mikroprocesor (Z80A) obrađuje istovremeno 8 bita (bajt) informacija, pa se naziva osmobitnim.

U memoriji se pamte programi i podaci. Sastoji se od integrisanih kola koja sadrže veliki broj memorijskih ćelija, od kojih se svaka sastoji od jednog ili više tranzistora. Osam ćelija čini jednu celinu (bajt). Svaki bajt ima svoj broj koji se naziva adresa.

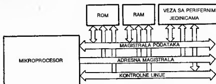
Najvažnije vrste memorija su RAM i ROM.

RAM mamorija (random access memory – memorija sa direktnim pristupom) je ona kod koje se može neposredno pristupiti bilo kojoj grupi memorijskih ćelija. U nju se podatak može upisati i iz nje pročitati. Sa nestankom napajanja podaci sa gube.

Iz ROM memorije (engl. read only memory) podaci se mogu samo čitati, a upisuju se u nju u procesu proizvodnje, tako da su trajno zabeleženi i ne zavise od napajanja.

Periferne jedinice služe za unošenje podataka (tastatura), njihovo prikazivanje (ekran, štampač) i masovno pamćenje (kaseta,

magnetna traka, magnetni disk, fleksibilni disk) Kola za vezu sa njima se zovu interfejsi (engl. interface) Spektrumova specifičnost je jedno kolo (ULA) u kome su integrisana kola za vezu sa perifernim jedinicama.



Slika 1–2 Sastav mikroračunara ZX SPEKTRUM

Pojedini delovi Spektruma su povezani magistralom podataka, adresnom magistralom i kontrolnim linijama.

Magistralu podataka (engl. data bus) čini osam vodova preko kojih se podaci (bajt) prebacuju iz memorije ili periferne jedinice u mikroprocesor ili obrnuto.

Adresna magistrala (engl. address bus) se sastoji od 16 linija preko kojih se određuje (adresira) sa kojim memorijskim ćelijama ili perifernim jedinicama se razmenjuju podaci. Može se adresirati 65536 (2^{16}) različitih lokacija (adresa).

Preko kontrolnih linija se kontroliše protok informacija.

1-4 PROGRAMSKI JEZICI

Naredbe koje mirkoprocesor izvršava su u obliku binarnih brojeva. Njih nazivamo mašinskim naredbama ili mašinskim instrukcijama. Skup svih takvih naredbi čini mašinski jezik. Program se može pisati i direktno na ovom jeziku, ali je to vrlo nepraktično.

Da bi se olakšalo programiranje, za svaku mašinsku naredbu je odabran simbol koji sa sastoji od nekoliko slova (najčešća je to skraćenica njihovog naziva na engleskom jeziku). Pojedine adrese i podaci takođe mogu da imaju svoja simbolička imena. Na ovaj način se došlo do simboličkog mašinskog jazika, odnosno asemblerskog jezika. Program se ovog jezika mora da se prevede na mašinski je-

zik da bi mogao da bude izvršen. Ovaj posao obavlja sistemski program koji se naziva asambler.

Viši programski jezici omogućavaju daleko lakše programiranje. Njihove naredbe zamenjuju niz mašinskih naredbi. Najčešće korišćani programski jezik je bajzik (BASIC) pomoću koga sa komunicira i sa Spektrumom.

Programe pisane na višim jezicima moguće je izvršavati na dva načina. Prvi je da se sve naradbe prevedu na mašinski jezik, i da se tako dobijeni program izvrši. Ovo se radi pomoću programa koji se zovu prevodioci ili kompajleri (engl. compiler). Drugi način je da se prevodi i izvršava naredba po naredba. Taj posao obavlja program čije je ime intarpreter. Na ovaj način programi se sporije izvršavaju. Sa Spektrumom se dobija bejzik interpratar, a mogu da se nabave pravodioci za bajzik i druga jazike.

1-5 KORISNIČKI I SISTEMSKI PROGRAMI

Krajnji cilj korisnika računara je da nešto izračuna, nacrta, uprevlja, obradi ili drugo, a to se postiže izvršavanjem korisničkog (aplikacionog) programa. Druga vrsta programa su sistemski programi čija je uloga da računar učina pristupačnim korisniku, tj. da omogu-ća da se napravi i izvrši korisnički program. Tu spadaju operativni sistemi, složeni programi koji predstavljaju programsku podršku računara tako da omogućavaju da se programi i podaci upišu u računar, prikažu, izmene, zepamta, da se programi puste u rad, zaustave i drugo. Za upisivanje teksta obično se koriste editori. Asembleri, prevodioci i intarpreteri takođe se svrstavaju u sistemske programe.

1-6 PROGRAMSKA PODRŠKA SPEKTRUMA

U Spektrumovom ROM-u, koji ima 16 Kbajta, nalaze se njegovi sistemski programi: operativni sistem i bejzik interpreter. Osim njih mogu se koristiti i drugi sistemski programi koji su na raspolaganju. To su najčešće aditori i kompajleri za druge programske jezike.

Korisničke (aplikacione) programe možemo praviti sami ili ih nabaviti gotove. Kada sami sastavljamo program, on je najčešće na bejziku. Na mašinskom jeziku je znatno teže programirati, pa se koristi samo kada je potrebna velika brzina rada i kada se štedi memorijski prostor. Programi koji se kupuju večinom su na mašinskom ja-

ziku, dok je na bejziku obično samo prvi deo koji služi za postavljanje početnih uslova i učitavanje sa kasetofona.

Kad se računar uključi, kontrolu nad njim ima operativni sistem (zabeležen je u ROM-u). On očitava stanje na tastaturi, ispisuje poruke na TV ekranu, učitava program sa trake i obavlja druge poslove tako da omogućava korišćenje sistema. Bejzik program izvršava bejzik interpreter oslanjajući se na operativni sistem, koji i dalje ima kontrolu nad računarom. Startovanjem mašinskog programa on preuzima kontrolu, a operativni sistem više nema uticaja

Uvod u rad sa Spektrumom

2

2-1 SASTAV MIKRORAČUNARSKOG SISTEMA SPEKTRUMA

Standerdni ZX Spektrum mikrorečunarski sietem se sastoji od mikroračunara ZX Spektrum, televizijskog prijemnika, kasetofone i mrežnog napajanja (ispravljača). Može se proširiti štampačem, mikrodrajvom, disk jedinicama, palicama za igru i drugim dodacima.

Spektrum se sastoji od štampane ploče se elektronskim komponentama i tastature.

Testatura služi za komandovanje računarom i unošenje programa i podataka. Ispravljač obezbeđuje 9V jednosmernog nepona potrebnih za nepejenje računara. Podaci, programi, poruke, crteži i drugo prikazuju se na TV ekrenu u boji. U samom Spektrumu se nalazi zvučnik, pe je moguće proizvesti tonove i melodiju. Podaci i programi se mogu pomoću kasetofona beležiti na megnetnoj traci (kaseti) ili sa nje prebacivati u računar.

2-2 PUŠTANJE U RAD

Minimalna konfiguracije potrebne za rad sestoji se od Spektruma se ispravljačem i TV prijemnika. Spektrum se povezuje sa televizorom preko kable dobijenog uz računar. Jedan njegov kraj se uključi u priključek na zadnjoj strani Spektruma koji je označen sa TV, a drugi u entenski priključak televizora. Ispravljač se uključi u gradsku mrežu (220V) i poveže sa Spektrumom preko priključka označenog sa 9VDC (engl. volts direct current – volti jednosmerne struje). Da bi se ne ekranu pojavila slika koju stvara računer, potrebno je da televizor podesimo ne 36. kanal (UHF opseg). Kada se na ekrenu pojavi poruka o autorskom prevu © **1982 Sinclair Research Ltd,** može se početi sa radom. Ukoliko se slika nije pojevila, proveriti ponovo da li je sve priključeno kako treba i da li je TV prijemnik dobro podešen.

2-3 TV EKRAN

Računar piša i crta po sradišnjem, pravougaonom delu ekrana. Obodnom dalu ekrana (bordar) možemo samo da odradimo boju, sem pri radu sa kasetofonom, kada se pojavljuju raznobojna pruge. Boja sradišnjeg dela akrana, kao i boja karaktera (slova, brojavi, znaci) i crteža, definiše se odgovarajućim naradbama.

Spektrum na akranu može da ispiše 24 rada sa po 32 karaktera. Odmah posle uključivanja i u toku večag dela vremena donjem delu ekrana pripadaju 2 rada koja sa nazivaju editorskim linijama. U njih se, pomoću tastature, upisuju naradbe ili podaci. Gornji dao ekrana sadrži 22 reda. Izmana u linijama gornjag dela ekrana moža sa izvršiti samo ako se one prebace u aditorske linije.

2-4 RAD SA TASTATUROM

Spektrum ima 40 tastara, od kojih svaki ima više funkcija. Jedna od njagovih specifičnosti je ispisivanja naredbi bejzika pritiskom na samo jadan tastar.

Kada sa računar uključi, na akranu se pojavljuje poruka **© Sinclair Research Ltd.** Pritiskom na taster **ENTER** pojavljuje se pokazivač (engl. cursor) u obliku slova K koje trepće. Sada se može početi sa upisivanjem. Kako će računar shvatiti pritisak na tastar, zavisi od načina rada, što je označeno oblikom pokazivača i od toga da li su pritisnuti tasteri **SYMBOL SHIFT** ili **CAPS SHIFT**.

Kada je pokazivač u obliku K (engl. kay word), pritisak na tastar sa slovom računar shvata kao naredbu ispisanu na tasterima balim slovima i ona se pojavljuje na ekranu. Pokazivač tada automatski dobija oblik slova L što znači da će pritisak na tastar sa slovima biti shvaćen kao malo slovo (engl. lowar casa).

Pokazivač se pojavljuje u obliku K na početku linije pre ispisivanja prve naredbe, posle naradbe **THEN**, i posle dve tačke kojima se označava početak nove naredbe u liniji.

PRIMER: Pokazivač je K, pritisak na tastar **P** daja **PRINT**, pokazivač prelazi u L, ponovni pritisak na tastar **P** daje **p**.

Kada je pokazivač u obliku K ili L, pritisak na taster sa brojem shvata sa kao broj.

Kada sa pritisne jadan od tastera dok je istovremano pritisnut **SYMBOL SHIFT**, to će biti shvaćeno kao crveno ispisana naradba ili znak na tastaru, baz obzira na to da li je pokazivač u K ili L obliku.

PRIMER: Dok je pritisnut tastar **SYMBOL SHIFT**, pritisak na taster **P** daje ".

Ako je pokazivač u L obliku, a pritisnut je **CAPS SHIFT**, pritisak na taster sa slovima će biti shvaćen kao veliko slovo.

PRIMER: Pokazivač je L, pritisnut je **CAPS SHIFT**, pritisak na taster **P** daje **P**.

Pritiskom na **CAPS SHIFT** i 2 **(CAPS LOCK)**, pokazivač menja oblik iz L u C (engl. capitals – velika slova) i pritisak na tastar sa slovom računar shvata kao valiko slovo i bez držanja tastera **CAPS SHIFT**. Dok je pokazivač u obliku C, pritiskom na **CAPS SHIFT** i **2 (CAPS LOCK)**, pokazivač pralazi u oblik L.

PRIMER: Pokazivač ja C, pritisak na taster **P** daje **P**.

Kada je pritisnut taster **CAPS SHIFT**, pritisak na taster sa brojam će biti shvaćen kao naredba ispisana belim slovima iznad tog tastera. One se odnose na upisivanje u računar i odmah se izvršavaju.

PRIMER: Dok je pritisnut **CAPS SHIFT**, pritisak na taster **0 (DELETE)** dovodi do brisanja onoga što je ispisano lavo od pokazivača.

Istovremenim pritiskom na tastere **CAPS SHIFT** i **SYMBOL SHIFT** pokazivač će dobiti oblik slova E (extended). Ponovnim pritiskom na oba tastera pokazivač dobija prethodni oblik. Kada je pokazivač E, pritisak na tastar će biti shvaćan kao zeleno ispisani simbol iznad njega, a ako se pri tome pritisne **SYMBOL SHIFT**, biće shvaćen kao crveno ispisani simbol ispod njega.

Pokazivač se iz E automatski vraća u prethodno stanje posle svakog pritiska na taster i ispisivanja simbola.

PRIMER: Pokazivač je E, pritiskom na tastar **P** dobija sa **TAB**
Pokazivač je E, pritisnut je **SYMBOL SHIFT**, pritiskom na taster **P** dobija se **©**.

Dok je pokazivač E, pritiskom na taster sa brojem određuje se boja osnove po kojoj će se ispisivati simboli na akranu **(PAPER)**. Ako je pokazivač E, a pritisnut je taster **CAPS SHIFT**, pritiskom na taster sa brojem određuje se boja zapisa na ekranu **(INK)**. Boja koja odgovara tasteru sa brojem naznačena je iznad odgovarajućeg tastera.

PRIMER: Pokazivač ja E, pritiskom na taster **2** nadalje će osnova **(PAPER)** biti crvene boje.
Pokazivač je E, pritisnut ja **CAPS SHIFT**, pritiskom na taster **4**, nadalje će zapis **(INK)** biti zelene boje.

Kada se pritisne **CAPS SHIFT** i **9 (GRAPHIC)**, pokazivač dobija oblik slova G. Vraćanje na prethodno stanje postiže se pritiskanjem tastera **G**. Kada je pokazivač G, pritiskom na tastere sa broje-

vima od 1 do 8 dobijaju se različito ispunjeni kvadrati (grafički karakteri). Ako je uz to pritisnut **CAPS SHIFT**, dobijaju se inverzni karakteri.

PRIMER: Pokazivač je G, pritiskom na **3** doboja se ▀.
Pokazivač je G, pritisnut je **CAPS SHIFT** ili **SYMBOL SHIFT**, pritiskom na **3** dobija se ▄.

Kada je pokazivač u G obliku, pritiskom ne tastere sa slovima dobija se grafika definisane korisnikom (ovo je objašnjeno u delu knjige o bejziku 3.2.).

Ako je taster pritisnut duže od sekunde, njegov simbol če biti ispisivan sve dok se taster ne pusti.

Brisanje se izvodi istovremenim pritiskanjem **CAPS SHIFT i Ø (DELETE)**. Briše se ono što je bilo levo od pokaziveča

U liniji koja se nalazi u donjem delu ekrana pokazivač se može kroz napisani tekst pomerati ulevo pomoću ⬅ **(CAPS SHIFT i 5)**, ili udesno pomoću ➡ **(CAPS SHIFT i 8)**. Ovo ne važi kada je pokazivač G ili E. Svrha pomeranja je da se pokazivač dovede na željeno mesto i tu upiše novi simbol ili izbriše neki stari (sa njegove leve strane).

Napomenimo još jednom da se naredba bejzika unosi samo jednim pritiskom na odgovarajući taster, dok je pokazivač K ili E.

PRIMER: Naredba **PRINT** se ne dobije pritiskom tastera **P, R, I, N** i **T**, nego pritiskanjem tastere **P** dok je pokazivač K.

2-5 NAČIN RADA

Spektrum može da radi na dva načina. Prvi je direktni ili kalkulatorski (engl. calculator mode), a drugi programski.

DIREKTNI

Kada se radi direktno, u donjem delu ekrane se ispisuje naredba, a zatim se pritiska taster **ENTER**. Tada računar odmah izvršava tu naredbu. Može postojati i više naredbi koje su medusobno odvojene sa dve tačke.

PRIMER: `CLS : PRINT "SPEKTRUM"`

Ovaj primer se unosi u rečunar na sledeči način: Dok je pokazivač u obliku K, pritisne sa taster **V**, zatim, istovremeno, **SYMBOL SHIFT i Z**, zatim **P** i ponovo zajedno **SYMBOL SHIFT i P**. Sada se napiše SPEKTRUM i na kraju pritisnu **SYMBOL SHIFT i P**.

Pritiskom na **ENTER**, računar če prvo izbrisati sve što je bilo na ekranu **(CLS)**, pa če zatim napisati u gornjem delu ekrana **SPEKTRUM**.

Ovaj način rada je pogodan za brza i kratka računanja u slučajevima da se rezultati neće dalje koristiti u računaru.

PRIMER Ako se napiše

```
PRINT 5.37*105+2.3↑(14/3.82 )
```

i posle toga pritisne **ENTER**, računar će izračunati ovaj izraz i napisati rezultat **585.01918**

NAPOMENA

Englezi upotrebljavaju decimalnu tačku, a ne decimalni zarez kao mi pa 5.37 odgovara našem 5,37
Simboli u prethodnom primeru su:
↑ simbol za stepenovanje
* simbol za množanja
/ simbol za deljenje.

Računar u diraktnom načinu rada ne pamti naredbe posle njihovog izvršavanja.

PROGRAMSKI

U programskom načinu rada prvo se piše program, a zatim se izvršava naredbom **RUN** i **ENTER.**

Program se sastoji od niza programskih linija. Svaka počinje brojem linije koji može da bude ceo broj od 1 do 9999. Za njim slede jedna ili više naredbi. Programska linija se upisuje u donji dao ekrana, a zatim se pritiskom na taster **ENTER** prebacuje u gornji dao ekrana, a istovremeno ja računar zapamti. Pri prebacivanju nove linije u gornji deo ekrana ona se svrstava među prethodno unete prema svom broju, tako da su sve linije poređane prema veličini svojih brojava.

Da bi se olakšalo ispravljanje programa i umatanje novih programskih linija, brojevi linija ne bi trebalo da budu uzastopni brojavi. Uobičajeno je da se razlikuju za 5 ili 10

PRIMER. Pišemo program na sladaći način:

```
10 CLS
20 PRINT "SPEKTRUM" .
15 PRINT 2+3
```

Vidi se da su linije u gornjem delu akrana porađana po veličini svojih brojeva.

Pokazivač linije se nalazi u jednoj od programskih linija, koja su u gornjem dalu ekrana, odmah posle broja linije i ima oblik znaka > Posle prebacivanja nove linije u gornji deo ekrana, pokazivač se nalazi u njoj. Može se prebacivati u liniju iznad pomoću ↑ **(CAPS**

SHIFT i 7) ili ispod pomoću ⬇ **(CAPS SHIFT i 6)**. Programska linija u kojoj se nalazi pokazivač može se prebaciti u donji deo ekrana radi nekih izmena pomoću **EDIT (CAPS SHIFT i 1).**

Programska linija se može izbrisati ako se otkuca njen broj i pritisne **ENTER**.

PRIMER: **10 ENTER** briša liniju broj **10** . Sada ja pokazivač linije nevidljiv. Pritiskom na tastera **CAPS SHIFT i 6** ⬇ dolazi u liniju 15 i ponovo postaja vidljiv.

Ako se prilikom pisanja nova programska linije ne poštuju pravila formiranja bejzik linije, posle pritiskanja tastera **ENTER** računar neće prebaciti liniju u gornji deo akrana, nego će staviti znak pitanja na mesto na kome je greška.

PRIMER `20 PRINT "SPEKTRUM`

Pokazivač treba komandama ⬅ **(CAPS SHIFT i 5)** i ➡ **(CAPS SHIFT i 8)** pomeriti do tog mesta i ispraviti grešku. U prethodnom primeru treba desno od SPEKTRUM staviti".

RUN

Kada se želi da računar izvrši program koji je napisan, pritiska se taster **RUN**, a zatim **ENTER**.

PRIMER: Prvo sa piše program

```
10 CLS
20 PRINT "SPEKTRUM"
```

Izvršava sa sa **RUN**.

Računar radi isto što i u primeru za direktni način rada, ali sada program može ponovo da se izvrši, jer je zapamćan. Ovaj program ja mogao da buda napisan i u obliku

```
10 CLS : PRINT "SPEKTRUM"
```

U prethodnom slučaju program ja izvršan od programske linije sa najmanjim brojem. Ako je potrabno, izvršavanje može da počne od linija koja se zada. Treba samo posla **RUN** napisati broj željene linije

PRIMER: Ako se u prethodni program doda linija

```
30 PRINT 2+3
```

i izvrši startovanje sa **RUN** i **ENTER**, na ekranu će biti napisano **SPEKTRUM**, a ispod toga **5**. Ako se program startuje sa **RUN 30** , na ekranu će biti napisan rezultat **5** jer je izvršena samo naradba u liniji **30** .

LIST

Od računara se može zatražiti da prikaže, odnosno, izlista upisani program. To se postiže naredbom **LIST**. Ako iza naredbe ne stoji nikakav broj, listanje počinje od linije sa najmanjim brojem, a ako je naveden broj, listanje počinje od linije sa tim brojem.

PRIMER: Ako se pritisne taster
LIST
računar će izlistati ceo program iz primera,
e ako se upiše
LIST 20
biće prikazan od linije 20

Pokazivač će se posle listanja nalaziti u liniji od koje je listanje počelo. Na ovaj način možemo kod dugačkih programa pokazivač prebacivati na željena mesta. Drugi način je da se unese broj neke nepostojeće linije blizak broju željene linije i pritisne **ENTER.** Pokazivač će se pojaviti pomoću komandi ⬇ ili ⬆

NEW

Program se briše iz memorije pomoću naredbe **.NEW**

PRIMER: Posle
NEW
računar će ispisati znak o autorskom pravu, kao pri uključivanju.
Ako se pritisne
LIST
vidi se da nema šta da izliste

BREAK

Pomoću naredbe **BREAK (CAPS SHIFT i SPACE)** prekida se izvršavanje programa.
PRIMER: Upisani program

```
10 CLS
20 PRINT "SPEKTRUM"
30 GO TO 10
```

startuje se sa **RUN i ENTER**. On će izbrisati sadržaj ekrana, napisati **SPEKTRUM**, zatim će se vratiti na liniju 10 i ponoviti postupak i tako još bezbroj puta. To se dešava veoma brzo, tako da se samo vidi kako netpis **SPEKTRUM** trepti. Naredbom **BREAK (CAPS SHIFT i SPACE)** izvršavanje programa će biti prekinuto.

CONTINUE

Prekinuti program će nastaviti da se izvršava pomoću naredbe **CONT** (CONTINUE).
PRIMER: Program iz prethodnog primera će nastaviti da se izvršava ako pritisnamo tastar **CONT.**

STOP

Ako u programu postoji naredba **STOP**, izvršavanja programa će biti prekinuto pri izvršavanju ta naredbe. Izvršavanje programa se može nastaviti naredbom **CONT.**
PRIMER: Program

```
10 PRINT "SPEKTRUM"
20 STOP
30 PRINT "********"
40 GO TO 10
```

će prilikom izvršavanja **(RUN)** napisati **SPEKTRUM** i zaustaviti se. Posle naredbe **CONT**, napisaće red zvezdica i vratiće se na liniju10, zbog čega će ponovo napisati **SPEKTRUM** i ponovo se zaustaviti. Postupak se može ponavljati.

SCROLL

Kada prilikom ispisivanja ne može sve da stana na ekran, računar će postaviti pitanje **scroll?** Pritiskom na taster **N, BREAK** ili **STOP**, ispisivanje će se prekinuti, a pritiskom na bilo koji drugi taster, računar će početi da pomera linije naviše, pri čamu će gornja biti izgubljene, a nove ća biti ubacivane odozdo. Ovo će se nastaviti dok se ne ispiše sve što je potrebno, ili dok se cao ekran ne popuni novim sadržajem, usled čega će se ponovo postaviti pitanje **scroll?**

PRIMER:
```
10 PRINT "SPEKTRUM"
20 GO TO 10
```

Uneti program daje računaru naredbu da napiša reć **SPEKTRUM,** pa da se vrati i ponovo izvrši tu naredbu. Izvršavanjem ovog programa **(RUN ENTER)**, na akranu ća biti ispisivano **SPEKTRUM** odozgo nadole sve dok se ne popune sva 22 reda. Zatim će na donjem delu ekrana biti ispisano scroll? i računar će čekati našu odluku. Pritiskom na taster **N** ili **STOP** (SYMBOL SHIFT i **A)** prakida se dalje izvršavanje, a pomoću bilo kog drugog tastera nastavlja se izvršavanje programa

IZVEŠTAJI

Kada prestane da izvršava bejzik program, računar u donjem delu ekrana ispisuje izveštaj kojim objašnjava razlog zaustavljanja. Izveštaj počinje slovom ili brojem koji predstavlja njegov kôd. Posle toga sledi kratak tekst poruke, a zatim dva broja. Prvi je broj linije u kojoj se program zaustavio, a drugi je broj naredbe u toj liniji pri kojoj se zaustavio program.

Sve vrste izveštaja su objašnjene u delu knjige 3.3.

Prilikom izvršavanja programa iz primera dobijale su se sledeće poruke: **0 OK**, što znači da je program izvršen bez problema, **9 STOP statement**, kada je program prekinut usled izvršavanja naredbe **STOP; D BREAK – CONT repeats,** kada je pritisnut **BREAK (CAPS SHIFT i SPACE)** i tada računar poručuje da pomoću naredbe **CONT** izvršavanje programa može da bude nastavljeno.

PRIMER

Pomoću programa iz sledećeg primera videće se gde i kako je u RAM memoriji računara zapamćen program koji je upisan u njega

PRIMER:

```
10 FOR n=23755 TO 24000
15 IF PEEK n<33 THEN  GO TO 30
20 PRINT n,PEEK n;TAB 26;CHR$
PEEK n
30 NEXT n
```

Izvršavanjem ovog programa na ekranu će se pojaviti tri kolone. U prvoj su neki od brojeva između 23755 i 24000. To su adrese u RAM-u (n) u kojima je smešten program. U drugoj koloni su brojevi (od 0 do 255) koji predstavljaju sadržaj memorije na tim adresama dat u decimalnom obliku **(PEEK n)**. Posmatrajući treću kolonu odozgo nadole videćemo da je to tekst programa pomešan sa još nekim znacima. On se dobija tako što računar prikazuje kako shvata sadržaj memorije, ako zna da je u pitanju tekst programa **(CHR$ PEEK n)**

Ovaj program ima oblik petlje. Naredbe koje se u njoj nalaze izvršavaju se više puta. U ovom slučaju to su naredbe u linijama 15 i 20. Početak petlje je linija 10, a kraj linija 30. Pri izvršavanju se promenljiva n menja u opsegu od 23755 do 24000, tako da se njena vrednost u svakom krugu povećava za jedan. Naredbe u petlji se izvršavaju 245 (24000 – 23755) puta

Programska linija 20 će nam u svakom krugu izvršavanja napisati vrednost n, adresu u RAM-u, zatim pola rada dalje broj koji se nalazi na toj adresi (taj broj će računar pretvoriti iz binarnog u decimalni i tako nam ga prikazati). U koloni 26 (**TAB** 26) prikazane su naredbe i znaci koji su predstavljani tim brojevima. Znaci koji nemaju smisla su razni kodovi za kontrolu načina upisivanja teksta, a ovde računar pokušava da ih prikaže kao znake jer je takva naredba (**CHR$**). Deo takvih kodova je izbačen pomoću naredbe u liniji 15.

2–6 UČITAVANJE PROGRAMA SA MAGNETOFONSKE TRAKE

Učitavanje programa pradstavlja prebacivanje programa ili raznih podataka sa magnatofonske traka (kasate) u računar. Da bi se to postiglo, treba povazati EAR priključak Spektruma sa EAR priključkom na kasetofonu i to jednim vodom dvožilnog kabla koji se dobija uz računar. Drugim vodom se povazuju MIC priključci Spektruma i kasetofona. Taj vod se koristi prilikom snimanja programe i na mora se spajati prilikom učitavanja.

Programe koji su prethodno upisani trebe izbrisati. To se postiže isključivanjem i ponovnim uključivanjem računara. Elegantnije rešenje je otkucavanje i izvršavanje naredbe **PRINT USR 0**.

Pošto se na ekranu pojavila poruka o autorskom pravu, unosi se naredba za učitavanja programa **LOAD ""**. To se postiže na sledeći način: pritisne se taster **J (LOAD)**, a zatim, dok je pritisnut taster **SYMBOL SHIFT**, dva puta se pritisne taster **P (")**. U slučaju greške, za brisanje se koristi taster **0** uz istovremeno pritisnut taster **CAPS SHIFT (DELETE)**. Pritiskom ne tastar **ENTER** sadržaj ekrana se briše i računar izvršava zadatu naredbu čekajući snimak programa.

Potenciometre za kontrolu jačine i, eventuelno, boje tona na kasetofonu treba postaviti u srednji položaj. Startovanjam kasetofona, obavezno sa trekom premotanom na početak programa, počinje učitavanje. Na obodnom delu slike (borderu) koji je do tada menjao boju iz crvenog u plavo i obratno, sada se iscrtavaju cveno-plave vodoravne linije u trajanju oko 5 sekundi. Odmah zatim u trajanju dela sekunde crveno-plave linije se zamanjuju tanjim plavo-žutim, a na ekranu se ispisuje poruka **Program:** ime programa. Taj početni deo se naziva zaglavlje (header) i nalazi se na početku svakog pro-

grama. Nosi informacije o imenu programa, njagovoj dužini i mestu započinjanja izvršavanja programa. Za zaglavljem sledi glavni deo programa koji se naziva blok i započinje crvenim i plavim linijama trajanja oko 2 sekunda. Nakon toga slede žute i plave linije u trajanju zavisnom od dužine programa. Program se sastoji od jednog ili više blokova kojima moža, a ne mora, prethoditi zaglavlje. Učitavši i poslednji blok, računar u donjem radu ispisuje poruku OK ili automatski otpočinje sa izvršavanjem programa, ispisujući na ekranu potrebne poruke za dalji rad. Kasetofon se zaustavlja i zatim se prelazi na rad sa programom.

Manji broj programa zahteva za učitavanje nešto izmenjenu naredu: **LOAD""CODE. CODE** se dobija pritiskom na taster **I** dok je pokazivač u obliku E. Ostali deo postupka učitavanja je isti kao u prethodnom slučaju, sem što se umesto poruke **Program:** ime programa, sada pojavljuje poruka **Bytes:** ima programa.

Vreme učitavanja programa može iznositi do četiri minuta. Za ispravno učitavanje potrabno je učitati cao program po tačno utvrdenom radosledu zaglavlja i blokova. Vodoravne linije na obodnom delu slike ukazuju da je učitavanje u toku. U slučaju graške pri učitavanju, računar moža, ali ne mora, ispisati poruku: **R: Tape loading error.** Tada je potrebno ponoviti cao postupak učitavanja, ali sa promenjanim položajem regulatora jačine i boje tona.

Sem u posebnim slučajevima, u računar se učitava isključivo po jedan program. Za učitavanje novog programa potrebno je obrisati prethodni.

Rad sa kasetofonom je detaljnije obraden u delu tri ove knjige u kome su opisana bejzik naredbe **SAVE, LOAD, MERGE** i **VERIFY.**

Bejzik 3

Bajzik (BASIC – Beginner's All-purposa Symbolic Instruction Code) je jedan od viših programskih jezika koji se danas najčešća koriste. Pogodan je za upotrebu u najfazličitijim oblastima (poslovne, naučne, tahničke itd.). Ono što ga čini privlačnim je činjanica da se lako uči i primanjuje. Broj formalnosti ja manji nego kod vaćina drugih jezika. Postoji veliki broj drugih programskih jazika koji su moćniji od bejzika, ali ja za njihovu upotrabu potrebna veće obučanost. Takođe, vaćina ih je specijalizovana za određanu vrstu primane.

U ovom dalu knjiga prikazaće sa Spektrumov bajzik. Izložana su njegova naredba, način njihove primene i pojmovi vezani za programiranja u bajziku.

3–1 OSNOVNI POJMOVI

KARAKTERI

Karaktari su znakovi koje Spektrum može da prikaže na ekranu ili štampaču. Obuhvataju slova (velika i mala), cifre, grafičke simbole, znakove interpunkcije, matematičke simbola, znakove koje je dafinisao korisnik i druga vrsta znakova

Kontrolni karakteri služe za kontrolu masta i načina ispisivanja. Označeni su sa jednim ili više karaktera.

IMENA NAREDBI

Svaka naradba ima ime koje se sastoji od niza karaktera. To su velika slova Iza kojih može da stoji još neki znak.

Svaki karakter, kontrolni karakter i naredba ima svoj osmobitni kôd i u računaru se pamti samo kao jedan bajt. Kada ja potrabno da sa ime naradbe prikaže na ekranu, odgovarajući kôd se interpratira kao niz karaktera (engl. token). Spisak kodova je dat na kraju knjige.

BROJEVI I BROJNE PROMENLJIVE

Za rad u bejziku na Spektrumu se koriste decimalni brojevi koji se predstvljaju nizom decimalnih cifara (izuzetak je pri upotrebi naredbe BIN). Njih odlikuje sledeće:

– umesto uobičajenog decimalnog zareza koristi se decimalna tačka

– nula se predstavlja ovim znakom Ø,

– prazna mesta između cifara nisu dozvoljena,

– broj može početi decimalnom tačkom,

– predznak se navodi ispred broja, a ako nije naveden, broj je pozitivan,

– broj se može predstaviti i u eksponencijalnom obliku.

PRIMER	pravilno	nepravilno
	3.14	3,14
	123ØØ	12 3ØØ
	.99	13.3ØØ
	123E2	13.3ØØ.5

Pozitivni brojevi mogu imati vrednost u opsegu od 2.94E–39 do 1.7E38, a isti je toliki i opseg negativnih brojeva. Njihova tačnost može biti 9 ili 10 cifara.

Brojni izraz je postupak za određivanje brojne vrednosti i sastoji se od argumenata (brojeva ili brojnih promenljivih) nad kojima se izvršavaju odgovarajuće operacije.

PRIMER: $\sqrt{4} + 3$

Za određivanje vrednosti navedenog izraza potrebno je izvršiti operaciju korenovanja argumenta 4, a zatim sabiranje argumenta 2 (rezultat prethodne operacije) i argumenta 3.

Brojna promenljiva je simbolička oznaka kojoj se može dodeliti brojni podatak. To se ostvaruje za to predviđenim naredbama (LET, INPUT, READ...). Promenljiva ne postoji (nije definisana) sve dok joj se ne dodeli vrednost.

PRIMER: **LET a = 3.14**

Brojna promenljiva ima ime a i tekuću vrednost 3.14.

Moguće joj je dodeliti i druge vrednosti.

Ime brojne promenljive je proizvoljno dugačke kombinacija slova i brojeva. Ime more počinjati slovom.

Nema razlike između velikih i malih slova. Upotreba drugih karaktera nije dozvoljena. Prazne meste (engl. space) i kontrolni karakteri boje mogu se sedržati u imenu, ali ga ne menjaju.

PRIMER	pravilno ime	nepravilno ime
	A	7C – počinje brojem
	B2A	$A3 – nedozvoljeni karakter
	QY EATENnGIh59	

Višedimenzionelnim promenljivama (vektori, matrice) potrebno je dodeliti prostor u memoriji neredbom **DIM.** Dimenzije i broj dimenzija nije ogrenićen. Ze imena višedimenzionalnih promenljivih važi isto što i za imena promenljivih, ali one mogu imati samo jedno slovo.

STRINGOVI I STRING PROMENLJIVE

String je niz karaktere (slove, brojeva i različitih vrsta znakova). Kod Spektruma dužina stringe nije ograničena.

Postoje takođe i string promenljive, čije ime se sestoji od samo jednog slova iza koga sledi znak $. Pri tome ne postoji razlike između velikog i melog slove

Kao i kod brojnih promenljivih, string promenljiva nije definisana dok joj se odgovarajućom naredbom (LET, INPUT, READ..) ne definiše vrednost.

Višedimenzionalne string promenljive se takođe definišu naredbom DIM, a broj dimenzija i dimenzije nisu ograničeni.

3–2 NAREDBE I NJIHOVA UPOTREBA

ENTER

Pri radu u direktnom (računarskom) režimu neredbom **ENTER** se nalaže računeru da izvrši nevedene naredbe iz editorske linije.

U programskom režimu je posle formiranja progremske linije potrebno de se one smesti u deo memorije u kome se pamti program. To se takođe postiže naredbom **ENTER.** Pored toga, programska linije se prebacuje u gornji deo ekrana. (**ENTER** zapravo ne

predstavlja bejzik naradbu već oparativnu naredbu. Isti je slučaj i sa naredbom **BREAK).**

BREAK

Naredbom **BREAK (CAPS SHIFT i SPACE)** prekida se izvršavanje tekućeg bejzik programa. Do prekida ća doći kada se izvrši tekuća naredba. **Na** donjam delu ekrana pojavljuja se izveštaj: **L BREAK Into program.** Naradbom **CONT** izvršavanja će biti nastavljano od sledeće naredbe.

Pri radu sa perifernim jedinicama i u slučaju da računar postavi pitanje **scroll?** do prekida dolazi pritiskom samo na taster **SPACE (BREAK).** Izveštaj u tom slučaju glasi: **D BREAK – CONT repeats. Naredbom CONT** izvršavanje se nastavlja od naredbe u kojoj je došlo do prekida

RUN

Kada je program unasan u memoriju, njagovo izvršavanje otpočinja naredbom **RUN.** Može se zadati direktno, ili, rede, u programu.

Naredbom **RUN** izvršavanje programa će otpočeti od programska linije sa najmanjim brojem. Stavljanjem celog broja iza naredbe **RUN,** izvršavanje programa otpočinje od programske linije sa tim brojem. Ako on ne postoji, izvršavanje otpočinje od prve naredne linija.

Broj linija se može zadati i u obliku promenljive ili izraza. Ako to nije ceo broj, biće zaokružen na najbližu calobrojnu vrednost.

Izvršavanjem **RUN** n (n je broj programska linija) brišu se sve prethodne programske promenljive i njen efekat je isti kao u slučaju naredbe **CLEAR: GO TO** n

NEW

Pre unošenja novog programa u računar treba pomoću **NEW** izbrisati iz RAM-a prethodni program i vrednosti programskih promenljivih. Sistemske promenljive treba postaviti na njihove početne vrednosti. RAM se briše naredbom **NEW** samo do lokacije na koju ukazuje sistemska promenljiva RAMTOP Takođe ostaju nepromenjene sistemske promenljive UDG, P RAMT, RASP i PIP Potpuno brisanja RAM-a se postiže isključivanjem računara ili izvršavanjem naredbe **GO TO USR0.**

LIST n

Naredbom **LIST** omogućava se prikazivanje prethodno unatog programa na akranu. Ako je iza **LIST** stavljen neki broj (n), prikazivanje otpočinje od linije sa tim brojem ili, ako ona ne postoji, od prve naredna. U slučaju da broj nije navadan, prikazivanja otpočinja od prve linije u programu.

Ako sa celokupni program ne moža prikazati na akranu, u donjam delu ekrana sa ispisuje poruka **scroll?** Tada se pritiskom na jedan od tastera: **N, STOP ili BREAK** prekida dalje prikazivanja programa, a pritiskom na bilo koji drugi taster prikazivanje se nastavlja.

U slučaju da broj iza **LIST** nija cao, biće zaokružan na najbliži ceo broj. Taj broj sa može zadati i pomoću izraza ili promenljivih.

Naradba **LIST** se po pravilu zadaje direktno mada je moguće njeno unošenje u program.

CONTINUE

Izvršavanje prekinutog programa može sa nastaviti naredbom **CONTINUE**. U slučaju da je izvaštaj o prakidu **L** ili 9, izvršavanja ća se nastaviti od sledeće naradbe. U ostalim slučajevima sa ponavlja naradba u kojoj ja izvršen prekid.

U slučaju prekida izvršavanja direktnih naredbi, nije moguće nastaviti izvršavanja.

STOP

Naredbom **STOP** zaustavlja sa dalja izvršavanje naredbi. U donjem dalu akrana ispisuja se izveštaj: **9 STOP statement** (videti tabelu sa izvaštajima). Naradbom **CONTINUE** nastaviće se izvršavanje programa od naradbe koja sledi iza naradbe **STOP**.

CLS

Naradba **CLS** briše sadržaj ekrana. Briša se sadržaj vidao memorija, ali ne i polja atributa.

REM

U cilju preglednosti dužih programa, naročito u toku njihovog razvijanja, potrebno je označavanje i objašnjavanje pojedinih mesta u programu. Za to služi naradba **REM**. Iza nja se mogu navoditi svi karaktari i naradbe, pe i naredbe razdvojene dvotačkom, ali se one neće izvršavati. Izuzetak ja **ENTER**.

PRIMER. `2940 REM Ulaz u potprogram 7`

Linijom 2940 iz nekog zamišljenog programa, prikazana je mogućnost **REM** naredbe da označi značajno mesto u programu.

Kada se razvijanje programa završi, **REM** naredbe se mogu ukloniti u cilju dobijanja kraćeg programa i bržeg izvršavanja.

PAUSE n

Izvršavanja naradbe **PAUSE** n traje n puta po 20ms (milisekunda = 1/1000 sekunde) ili do pritiskanja nekog tastera, posle čega se prelazi na izvršavanje sledeće naradbe. Broj n je iz opsega od Ø do 65535 i može se uneti kao promenljiva ili izraz.

Ako je n nula, pauza nije vramenski ograničena i traje sve dok sa na pritisna neki taster, što se veoma često koristi. Primer dat za naredbu **INKEY$** prikazuje način njenog korišćanja.

Upotreba **PAUSE** se ne preporučuje za generisanje tačnih vremenskih parioda. Trajanje prvog od n perioda je nedefinisano dok je ostalih n-1 perioda po 20 ms.

Izvršavajući ovu naredbu Spektrum broji poluslike koje prikazuje na TV akranu.

PRINT

Ovom naredbom se na ekranu ispisuju rezultati izračunavanja, tekst i različiti karakteri.

Način upotrebe naredba **PRINT:**

1. Ispisivanje brojeva, vradnosti brojnih promenljivih i izraza.

PRIMER:
```
10 FOR n=-11 TO 12: PRINT n,9↑
n: NEXT n
```

Ovim primerom se ispisuju vrednosti promenljive n i broja 9 dignutog na n-ti stepen, pri čamu se n menja od – 11 do 12.

2. Ispisivanje stringova i vrednosti string promenljivih i izraza sa stringovima. Ispred i iza stringa treba staviti navodnike.

PRIMER	naradba	ekran
	PRINT "1985. god."	**1985. god.**
	10 LET a$ = "SPEKTRUM": PRINT a$	**SPEKTRUM**

3. Pomoću kontrolnog karaktera može se odrediti mesto ispisivanja onoga što iza njega sledi

– apostrof ' sledeći red
– zarez , pola reda iza
– tečka zared ; odmah iza

PRIMER:

naredba	ekran
PRINT 1,2,3,4	**1 2**
	3 4
PRINT 1;2;3;4	**1234**
PRINT 1'2'3'4	**1**
	2
	3
	4

4. Ako se iza naredbe PRINT ne navede ništa, pri izvršavenju sledeće **PRINT** naredbe ispisivanje će otpočeti jedan red niže.

PRIMER:

naredba	ekran
PRINT 1: PRINT :	**1**
PRINT 2	
	2

5. Kontrolni karakter AT omogućava ispisivanje na željenom mestu ekrana. Naredba tada ima oblik

10 PRINT AT x,y; ono što se ispisuje

x je red, a y kolona u kojoj se sipisuje. Redovi i kolone se broje od gornjeg levog ugla. x može biti od 0 do 21, a y od 0 do 31.

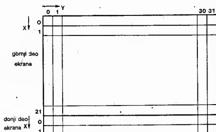
PRIMER:
```
10 PRINT AT 7,15;"+"
```

Zn + se ispisuje u sedmom redu i petnaestoj koloni.

PRIMER:
```
10 FOR n=0 TO 15: PRINT AT n,2
*n;"+": NEXT n
```

Promenljiva n se menja od 0 do 15. Znak + se ispisuje u redu n i koloni 2n.

Brojevi x i y koji nisu celi zaokružuju se na najbližu celobrojnu vrednost. Negativni brojevi se tretiraju kao pozitivni.



Slika 3–1 Koordinatni sistemi na ekranu

6. Kontrolni karakter **TAB** omogućava ispisivanje u željenoj koloni. Naredba traba da ima oblik

PRINT TAB x; ono što se ispisuje

i dovodi do ispisivanja u koloni x tekućeg reda. x je broj o 0 do 31.

PRIMER:
```
10 PRINT AT 3.9;"logicko stanj
e"
20 PRINT "ulaz A";TAB 11;"ulaz
B";TAB 25;"izlaz "
30 PRINT "
-----------------------------"
```

Na ovaj način formiramo zaglavlje tabele.

7. Kontrolni karakter # omogućava ispisivanje u donjam delu ekrana. Upotrebljava se u obliku

PRINT # 0;AT x,y; ono što se ispisuje

gde je x red, a y kolona brojano od gornjeg levog ugla donjeg dala ekrana.

8. Naredbe boje i ispisivanja **PAPER, INK, FLASH, BRIGHT, INVERSE** i **OVER** mogu se upotrebljavati iza naredba **PRINT.**

LET

Promenljivoj se vrednost može dodeliti upotrebom naredbe **LET.** To ee može uraditi i pomoću naredbi **INPUT** i **READ.** Promenljiva nije definisana sve dok joj se ne dodeli vrednost. Dodeljenu vrednsot zadržava sve dok joj se ne dodeli nova.

– BROJNE PROMENLJIVE

Dodeljivanje vrednosti brojnoj promenljivoj ostvaruje se naredbom u obliku.

LET a=b

gde je a ime promenljive, e b vrednost koja joj se dodeljuje. b može biti broj, brojna promenljiva ili izraz.

PRIMER:
```
10 LET a=3
20 LET q1=a
30 LET abc12=a+SQR q1
40 PRINT a'QL'abc12
```

U liniji 10 promenljive a dobije vrednsot 3, u liniji 20 promenljiva ql dobija vrednost promenljive a, u liniji 30 promenljiva abc12 dobije vrednost zbira vrednosti promenljive e l kvadratnog korena promenljive ql, a naredbom u liniji 40 se lspisuju vrednosti promenljivih a, ql i ebc12.

– STRING PROMENLJIVE

Dodeljivanje vrednosti string promenljivoj ima isti oblik kao u slučaju brojne promenljive. String promenljivoj se može dodeliti vrednost u obliku stringa, string promenljive ili izraza sa stringovima.

PRIMER:
```
10 LET a$="Mikro"
20 LET b$=a$
30 LET a$=b$+" Knjiga"
40 PRINT a$
```

U liniji 10 se string promenljivoj a$ dodeljuje string Mikro. Ispred i iza stringe su obavezni znaci navoda. U liniji 20 promenljivoj b$ se dodeljuje vrednost promenljive e$, a u liniji 30 promenljiva a$ dobija vrednost koja je zbir vrednosti promenljive b$ i stringa knjiga. Neredbom u liniji 40 ispisuje se vrednost a$.

Dodeljivanje vrednosti višedimenzionalnim promenljivama je prikazano u opisu naredbe **DIM**.

INPUT

Pomoću naredbe **INPUT** promenljivama se dodeljuje vrednost direktno preko tastature od strane korisnika. Izvršavajući ovu naredbu računar čeka na unošenje podataka. Pokazivač se postavlja u donji deo ekrana, gde se ti podaci ispisuju. Unošenje se završava pritiskom na taster **ENTER**.

Brojnoj promenljivoj vrednost se može dodeliti unošenjem broja, brojne promenljive ili brojnog izraze. String promenljivoj vrednost se daje unošenjem stringa (teksta, odnosno niza karaktera). U ovom slučaju računar automatski sa obe strene pokazivača, odnosno, stringa koji se unosi, postavlja znakove navoda.

PRIMER:

```
10 INPUT a,b$
20 PRINT a;b$
```

Računar prvo očekuje unošenje broja, a zatim unošenje stringe. Posle unošenja, oni će biti ispisani na ekranu naredbom u liniji 20.

Naredba **INPUT** omogućava da se na ekranu, pre unošenja podataka, ispiše ime promenljive koja se unosi, ili bilo kakav drugi tekst. To se postiže tako što se taj tekst pod znacima navoda stavlja iza naredbe. Kontrolni karakteri: zarez (,), tačka zarez (;) i apostrof (') imaju isto značenje kao kod neredbe **PRINT**.
Ispisivanje se vrši u donjem delu ekrana.

PRIMER:

```
 10 REM izracunavanje predjenog
puta za zadatu brzinu i vreme
 20 INPUT "brzina [km/h]",v,"vr
eme [h]",t
 30 PRINT
 40 PRINT "predjeni put je ";v*
t;"km"
 50 GO TO 20
```

PRIMER: dodela vrednosti string promenljivoj

```
10 INPUT "Moje ime je",a$
20 FOR n=0 TO 10
30 PRINT AT n,n;a$
40 NEXT n
```

String je moguća unati i baz znakova navo... To se postiža dodavanjem naredbe **LINE** ispred imena string promenljiva unutar naradbe **INPUT.**

PRIMER: Linija 10 u gornjam primeru moža biti zamanjena sa:

```
10 INPUT "Moje ime je", LINE a
$
```

Kada se naradbom **INPUT** unosi string, moguće je pomoću **EDIT** ili **DELETE** izbrisati znakove nevoda koji se nalaze sa obe strane pokazivača. Tada se stringovi mogu unositi kao string promenljive ili izrazi sa stringovima.

PRIMER:
```
10 LET a$="POBEDA"
20 INPUT x$
30 PRINT x$
```

Brisanjem znakova navoda i upisivanjem stringa a$ ispisaće se vrednost stringa.

Podaci se prilikom unošenja pomoću naredbe **INPUT** mogu ispisivati u bilo kom delu ekrana. To sa postiža tako što sa proširuje broj redova koji pripadaju donjem dalu ekrana pomoću kontrolnog karaktera **AT**, a zatim sa, takode pomoću **AT**, zadaje masto ispisivanja u odnosu na novi koordinatni početak donjeg daia ekrana. Naredbom **INPUT AT x,0; AT m,n;a** početak donjeg deia ekrana će sa pomeriti x redove naviše. Prilikom unošenja promenljiva a će biti ispisana m radove nadole u koloni n u odnosu na novi koordianti početak.

PRIMER: Formiranja i popunjavanje jedna tabale

```
10 PRINT TAB 3;"x";TAB 14;"x↑2
";TAB 25;"x↑3"
20 PRINT "____________________
____________"
```

```
30 FOR n=0 TO 9
40 INPUT AT 20-n,0;AT 0,3;x
50 PRINT AT 2+n,3;x;TAB 14;x↑2
;TAB 25;x↑3
60 NEXT n
```

Program se prilikom izvršavanja naredbe **INPUT** ne može prekinuti pomoću naredbe **BREAK,** već pomoću naredbe **STOP.** Tada je izveštaj **H STOP In INPUT.** Kada se unose stringovi, treba prvo pomoću DELETE izbrisati levi znak nevoda.

Kada sa izvršava naredba **INPUT** u kojoj se nalazi naradba **LINE,** program se zaustavlja sa **↓ (CAPS SHIFT i 6).**

INKEY$

Prilikom izvršavanja naredbe **INKEY$** računar utvrđuje koji taster je u tom trenutku pritisnut. Rezultat je string koji sadrži karakter pritisnutog tastera, ili prazan string ukoliko nijedan taster nije pritisnut. Karaktari koji se na ovaj način dobijaju isti su kao i oni koji se ispisuju preko tastatura kada je pokazivač **L** ili **C.**

Za razliku od naredba INPUT, odmah nakon očitavanja tastature prelazi sa na izvršavanje sledeće neredbe, bez obzira da li je neki taster pritisnut ili na.

PRIMER: Sledaći program omogućava ispisivanje teksta kao na pisaćoj mašini

```
10 IF INKEY$<>"" THEN GO TO 1
0
20 IF INKEY$="" THEN GO TO 20
30 PRINT INKEY$;: GO TO 10
```

U liniji 10 se nalaže računaru da čeka dok se prethodno pritisnuti taster ne pusti, a u liniji 20 da čeka dok neki taster ne bude pritisnut.

PRIMER: Naredni program ispisuje karaktere pritisnutih tastera, sem kada se unosi f.

```
10 LET a$=INKEY$
20 IF a$<>"f" THEN PRINT a$
30 PAUSE 0: GO TO 10
```

Zamenom linije 20 linijom:

```
20 IF a$="f" THEN PRINT a$
```

prilikom pritiskanja tastera, karakter će biti ispisan samo eko je to f.

READ, DATA

Pomoću ove dve naredbe promenljivame se dodeljuju vrednosti.

Iza naredbe **READ** nevode se promenljive kojima se dodeljuje vrednost. Medusobno su odvojene zarezima. Iza naredbe **DATA** navode se vrednosti koje se dodeljuju promenljivame i one su takode odvojene zarezima.

PRIMER:

```
10 DATA 7,"AVION",-2,16
20 READ a,b$,c
30 PRINT a'b$'c
```

Prvoj promenljivoj koja je navedena iza naredbe **READ** dodeljuje se prve vrednost iza neredbe **DATA**, drugoj promenljivoj ize **READ**, druga vrednost iza **DATA** i tako redom. Brojnoj promenljivoj treba dodelitl broj, a string promenljivoj string. U protivnom računar će javiti grešku sa izveštajem **C**.

Naredba **DATA** može da se nalazi na bilo kom mestu u programu. One se ignoriše sve do izvršavanja neredbe **READ**.

U slučaju de je broj promenljivih veći od broje podataka, javlje se greška sa izveštajem **E**.

Jedna naredba **READ** može dodeljivati promenljivama podatke pomoću više neredbi **DATA**. Kade završi sa dodeljivanjem vrednosti iz jedne naredbe **DATA**, prelazi ne vrednosti iz sledeće naredbe.

PRIMER:

```
10 READ a,b$,c$
20 PRINT a'b$'c$
30 DATA 1,"DA"
40 DATA "NE"
```

Promenljivama koje se nalaze iza više naredbi **READ** mogu da budu dodeljene vrednosti navedene iza jedne naredbe **DATA**. Prve neredba **READ** koristi prvi deo nevedenih vrednosti, a druga nared-

ba **READ** sledeće vrednosti i tako redom (sem kada postoji naredba **RESTORE**).

PRIMER:

```
10 READ a,b$
20 READ c
30 PRINT a'b$'c
40 DATA 1,"KNJIGA",2
```

PRIMER: Prikazan je jedan način da se elementima matrice dodele vrednosti.

```
10 DIM a(3,4)
20 FOR n=1 TO 3: FOR m=1 TO 4
30 READ a(n,m)
40 PRINT AT 3*n,3*m;a(n,m)
50 NEXT m: NEXT n
60 DATA 15,6,19,34,4,28.0,56,7
,10.22,43
```

RESTORE n

RESTORE omogućava da vrednosti navedene iza **DATA**, koje su već dodeljene promenljivama, budu ponovo dodeljene nekom drugom skupu promenljivih.

Ponovno dodeljivanje vrednosti se može obaviti samo iz onih naredbi **DATA** koje se nalaze u programskim linijama čiji je broj veći ili jednak broju koji je naveden iza **RESTORE**.

PRIMER:

```
10 DATA 7,"Avion",-2,16
20 DATA 123,"Knjiga"
30 READ a,b$,c,d,e$
40 PRINT a'b$'c'd'e$
50 RESTORE 15
60 READ a,b$
70 PRINT '''a'b$
```

Naredba **READ** u liniji 60 ponovo koristi podatke iz linije 20.

GO TO n

Naredbom **GO TO** n se prelazi na izvršavanje naredbi u programskoj liniji n. Ukoliko navedena linija ne postoji u programu, prelazi se na prvu sledeću. Broj n može biti u obliku promenljive ili izraza, a ako nije ceo broj biće zaokružen na bližu celobrojnu vrednost.

PRIMER:
```
10 LET x=1
20 PRINT x: LET x=x+1
30 GO TO 20
```

Izvršavanjem ovog programa na ekrenu se ispisuje kolona brojeva počevši od 1. Prethodni primer ime praktičnu primenu pri formiranju kašnjenja, naročito u programima koji se koriste pri merenjire i upravljanju.

GOSUB n, **RETURN**

U programima se često javlja potreba de se grupe naredbi izvršava više puta. Da se svaki put, kada je to potrebno, taj deo programa ne bi ponavljao, on se izdvaja i naziva potprogramom. Naredbe koje se nalaze u potprogramu izvršavaju se pomoću naredbi **GOSUB** n, gde je n broj linije u potprogremu. Ukoliko linije sa brojem n ne postoji, izvršavanje počinje od prve sledeće.

Posiednja naredba u potprogramu je uvek **RETURN**. Njenim izvršavenjem se vraća iz potprograma i počinje de se izvršava prva naredba posle **GOSUB**.

PRIMER:
```
10 LET a$=INKEY$
20 IF a$="b" THEN  GO SUB 200
30 GO TO 10
200 FOR n=0 TO 7
210 BORDER n: PAUSE 50
220 NEXT n: RETURN
```

Kada se izvršava ovaj program, priiikom pritiska na taster **b** skočiće se na potprogram koji počinje na liniji 50 i tada će se na obodnom delu ekrana menjati boje.

Broj linije dat iza naredbe **GOSUB** može biti u obliku brojnog izraze ili promenljive.

Naradba **GOSUB** se razlikuju od naradba **GO TO** po tome što se pamti broj linije i naredba sa koje se prešlo na izvršavanje potprograma, čime se omogućava povratak na sledeću naredbu po završatku potprograma. Ti podaci se pamte u delu mamorije koji se naziva **GOSUB** stek

Iz potprograma je moguće pozivati druga potprograma.

Prilikom pravljenja velikih programa preporučuje se da se oni izdele na manje celine, module. Time se dobija na preglednosti i olakšava programiranje i tastiranje. Ako se ti moduli pozivaju više puta, oni se prave u obliku potprograma.

IF THEN

Ova naredbe omogućava grananje u programu, u zavisnosti od toga da li ja uslov navaden u naradbi ispunjan ili nije. Njen opšti oblik je:

IF uslov **THEN** naredba

Ako je uslov ispunjen, nastavlja se sa izvršavanjem naredbi navadenih iza **THEN**, uključujući sve naredbe do kraja progremske linije. Ako uslov nije ispunjen, prelazi se na izvršavanja naredbi u sladećoj progremskoj liniji.

Uslov moža da bude rezultat operacije poradjenja (=, <, >, < =, > =, < >) ili rezultat logičke oparacije (NOT, AND, OR). U poredanjima i logičkim operacijama mogu se koristiti brojna i string promenljive i izrazi.

PRIMER:

```
10 LET a$=INKEY$
20 IF a$="a" THEN  PRINT a$;
30 GO TO 10
```

Pomoću ovog programa se očitava stanje na tastaturi. Ako je pritisnut tastar a (uslov a$ = "e"), na ekranu se ispisuje a.

IF THEN omogućava formiranje petlje u programu (grananje i petlja spadaju u osnovna programske strukture)

PRIMER

```
10 LET a=0
20 PRINT a,a*a
```

```
30 LET a=a+1
40 IF a<11 THEN GO TO 20
50 STOP
```

Program ispisuje vrednosti a i a² (linija 20) kada se a menja od 0 do 10. Pomoću linije 30 vrednost a se uvećava za jedan, a naredba u liniji 40 dovodi do ponavljanja prethodnih naredbi sve dok je a < 11.

PRIMER:

```
10 PRINT "Želite li ovaj prime
r ?"
20 PRINT "Odgovorite sa da ili
ne"''
30 INPUT a$
40 IF a$<>"da" THEN PRINT "Ba
s ste lenji"
50 IF a$="da" THEN PRINT "SUP
ER"
```

Ako je odgovor bio da, treba izbrisati prethodni primer i uraditi sledeći. To je jedna mala igra brzine reagovanje.

PRIMER:

```
10 LET a$="G"
20 LET b$="GRRRRRRRRRRRRRRRRRR
RR"
30 PRINT AT 10,2;b$
40 LET a$=a$+"R"
50 PRINT AT 11,2;a$
60 IF INKEY$="" THEN GO TO 40
70 IF a$<b$ THEN PRINT AT 5,1
0;"Kratko": STOP
80 IF a$>b$ THEN PRINT AT 5,1
0;"Dugo": STOP
90 PRINT AT 5,10;"ODLICNO"
```

Posle startovanja programe, pritiskom na bilo koji taster trebe zaustaviti ispisivanje, tako de ono što se ispisuje bude iste dužine kao i ono što je već nepisano.

FOR, TO, STEP, NEXT

Petlja je jedna od osnovnih programskih struktura. Koristi se kade jednu ili više naredbi treba izvršiti više puta redom.

U Spektrumu se petlja može formirati pomoću naredbi **FOR** i **NEXT. FOR** je prva naredba u petlji, koja ima opšti oblik

FOR n=e **TO** b **STEP** c

n je ime promenljive koja se naziva indeks petlje. Ona se u svakom krugu izvrševanje menje onako kako je to definisano ostlaim brojnim vrednsotima. Indeks se sastoji od samo jednog slove a je početna vrednsot indeksa, b je krajnja vrednsot indeksa, a c je korak promene. Ova tri broje ne moraju da budu cela, a mogu biti i negativna. Takođe mogu biti data i u obliku promenljivih i brojnih izraza. Ako se **STEP** i broj c izostave, korak promene je 1.

Poslednja naredba u petlji ima oblik **NEXT n.**

PRIMER:

```
10 FOR n=0 TO 10 STEP .5
20 PRINT n,n*n
30 NEXT n
```

Pri izvršavanju ove petlje indeks n se menja od 0 do 10 sa korakom 0.5. U svekom krugu izvršavenja ispisuju se vrednsoti n i n^2.

PRIMER: Ako se linija 10 izmeni u

```
10 FOR n=0 TO 10
```

korak promene n će biti 1.

Ako je početna vrednost indeksa menje od krejnje vrednosti, e korak promene negativan, ili eko je početna vrednost indekse veće od krajnje vrednosti, a korak promene pozitivan, petlja se neće izvršavati, već će se preći na izvršavanje sledeće naredbe iza **NEXT.**

Petlja se može napustiti i pre nego što indeks dostigne krajnju vrednost. Takođe je dozvoljeno i vraćanje u petlju. Pri tome trebe obratiti pažnju na vrednosti indeksa. Oni se mogu menjati van petlje, ali trebe biti oprezen.

Petlja se često koristi i za formiranje kešnjenje.

PRIMER:

```
10 FOR n=0 TO 100
 20 PRINT AT 5,5;n: BEEP .1,10:
BEEP .1,0
 30 IF INKEY$="s" THEN GO TO 1
00
 40 NEXT n: PRINT AT 10,10;"Zav
rsio sam    ": STOP
 100 PRINT AT 10,10;"   Stao sam
   "
 110 FOR j=1 TO 200: NEXT j
 120 PRINT AT 10,10;"ali ne za d
ugo": GO TO 40
```

U ovom programu postoje dve petlje. U prvoj se njen indeks n menja od 1 do 100. Pri tome se ispisuje vrednost indeksa i proizvode zvučni efekti. Pritiskom na taster S petlja se napušta i ulazi se u drugu sa indeksom j, pomoću koje se ostvaruje peuza. Po njenom završetku vraća se u prvu petlju. Osim ovog rasporeda petlji moguć je raspored u kom se jedne petlje nalazi u drugoj – dvostruke i višestruke petlje.

Pri upotrebi dvostrukih i višestrukih petlji, one se ne smeju preklepati, već jedna mora biti u drugoj, tj. početak i kraj jedne petlje moraju biti između početka i kraja druge.

PRIMER:

```
10 FOR i=0 TO 7
20 FOR n=0 TO 21
30 PRINT AT n,4*i;i*22+n
40 NEXT n
50 NEXT i
```

Petlja sa indeksom n se nalazi u petlji sa indeksom i. Na ekranu se štampa matrica brojeva.

DIM

Naredbom **DIM** se obezbeđuje prostor u memoriji za smeštanje vrednosti višedimenzionalnih brojnih i string promenljivih (vektora i matrica). Iza naredbe **DIM** se navodi ime, a zatim se u zagradi daju dimenzije te višedimenzionelne promenljive

PRIMER: **DIM e(5)**

Ovom naredbom se definiše vektor (jednodimenzionalna promenljive) sa imenom e, čiji su elementi promenljive e(1), e(2), e(3), e(4) i e(5).

PRIMER: **DIM f$(6)**

Ova naredba definiše vektor čiji su elementi šest string promenljivih.

PRIMER: **DIM g(3,4,5)**

Na ovaj način definišemo trodimenzionalnu matricu koja sadrži 60 promenljivih. One su označene se:

g(1,1,1), g(1,1,2), ...,g(1,1,5)
g(1,2,1), g(1,2,2), ...,g(1,2,5)
:
g(2,1,1), g(2,1,2), ...,g(2,1,5)
g(2,2,1), g(2,2,2), ...g(2,2,5)
:
g(3,4,1), g(3,4,2), ...,g(3,4,5)

Izvršavanjem naredbe **DIM**, sve promenljive, elementi definisane višedimenzionalne promenljive, dobijaju vrednost 0 eko se radi o brojnim promenljivama, odnosno postaju prazan string, eko se radi o string promenljivama. Za svaku brojnu promenljivu se rezerviše 5 bajta u memoriji, a jedan bajt za svaki karakter string promenljive.

Dimenzije višedimenzionelne promenljive mogu biti zadate i u obliku promenljivih i izraza, a eko nisu celi brojevi, zaokružuju se na najbliži celobrojnu vrednost.

Promenljivama, koje su elementi vektora i matrica, vrednost se dodeljuje uobičajenim naredbama za dodelu vrednosti. Pri tome je potrebno u zagradama navesti indekse promenljive koji su međusobno odvojeni zarezima.

PRIMER: **LET h(2,3,5) = 10**

Pomoću ove naredbe se elementu trodimenzionalne matrice h, čiji su indeksi 2,3 i 5 dodeljuje vrednost 10 (peti element u trećoj koloni i drugoj vrsti).

Svi elementi višedimenzionalne string promenljive su iste dužine. Ta dužina je jednaka poslednjoj nevedenoj dimenziji.

PRIMER:

```
10 DIM b$(2,6)
20 LET b$(1)="OLOVKA"
30 LET b$(2)="PAPIR"
```

U liniji broj 10 je definisana jednodimanzionalna string promenljiva (vektor), koja se sastoji od dve promenljive. Svaka od njih je string dužine 6 karaktera. Narednim naredbama **LET** se tim promenljivama dodaljuje vrednost.

U slučaju da se promenljivoj dodali string manje dužina od definisana, preostali deo do dafinisane dužine se popunjava praznim mastima. Ako karakter ima više od dafinisanog broja, oni koji su višak izostavljaju se. U gornjem primeru promenljiva b$(2) ća kao poslednji, šesti karaktar, imati prazno mesto (space).

Višedimenzionalne string promenljive možemo posmatrati i kao matrice, čiji svi elementi imaju samo jedan karakter, dok su dimenzije matrice svа dimanzije navedena u naredbi **DIM**. Svakom elementu matrice, odnosno svakom karakteru, možemo dodeljivati vrednost, ili ga upotrebljavati u raznim naredbama.

PRIMER. Višedimenzionalnu promenljivu b$ iz gornjeg primera možemo posmatrati kao dvodimenzionalnu matricu, čiji elementi sadrže po jedan karakter. Naredba

LET b$(2,4)="E"

će promeniti četvrti karakter u reči PAPIR iz I u E, a naredba

PRINT b$(1,5)

bi dovela do ispisivanja karaktera V (prva reč, peti karakter).

PRIMER.

```
10 DIM a$(3,4)
20 FOR n=1 TO 3: FOR m=1 TO 4
30 READ a$(n,m)
40 PRINT AT 3*n,3*m;a$(n,m)
50 NEXT m: NEXT n
60 DATA "a","b","c","d","exy",
"f","g","h","i","j","k","l"
```

Prvom naredbom se definiše višedimenzionalne string promenljiva a$. Možemo smatrati da je to vektor koji se sastoji od tri stringa sa po četiri elemenete, ili da je to matrica sa tri reda od po četiri kolone karaktere. U programu je formirana dvostruka petlja pomoću koje se svakom elementu matrice dodeljuje kerakter, e odmah zatim on se ispisuje. String exy će biti skrećen na e, jer svaki element matrice može sadržati samo jeden karekter.

ARITMETIČKE OPERACIJE

Spektrumov bejzik omogućave eritmetičke operacije:

+ sabiranje
− oduzimanje
* množenje
/ deljenje

Ove operecije se obevljaju između dve broja, argumente. Brojevi mogu biti zadati u obliku promenljivih i izraze.

Množenje i deljenje imeju viši prioritet, pa će u izrazime biti izvršeni pre sabiranja i oduzimanja.

PRIMER:
```
10 LET x=2: LET y=3
20 PRINT x+y'x-y'x*y'x/y
```

FUNKCIJE

Spektrumov bejzik omogućeva sledeće funkcije:

SQR x, kvadratni koren ($\sqrt{x}$); $x \geq 0$

Za $x < 0$ javlja se greške sa izveštajem: **A Invalid argument** (pogrešen argument).

PRIMER:
```
10 PRINT SQR 2
```

LN x prirodni logaritam tj. logeritem za osnovu e(lnx); $x > 0$ inače se jevlje greška sa izveštajem **A Invalid argument.** Pomoću ove funkcije se može odrediti vrednsot logaritma za bilo koju osnovu pomoću izraza

$\log_a x = \ln x / \ln a$

Ze dekadni logaritam: $\log_{10} x = \ln x / \ln 10$

PRIMER.
```
10 PRINT LN 9'LN 9/LN 10
```

x ↑ y, stepenovanja (x^y)

PRIMER: `10 PRINT 2↑3`

EXP x, eksponencijalna funkcija (e^x)

PRIMER: `10 PRINT EXP 4`

INT x, zaokruživanja na manji cao broj

PRIMER: `10 PRINT INT 3.7'INT -4.1`

Obratiti pažnju da je **INT** – 4.1 = –5.

ABS x, apsolutna vrednost |x|

PRIMER: `10 PRINT ABS -9.3'ABS 0'ABS 9`

SGN x, signum (sgn x)

sgn x = 1; x>0
sgn x = 0; x=0
sgn x = –1: x<0

PRIMER: `10 PRINT SGN -9.3'SGN 0'SGN 4.9`

PI funkcija bez argumenta koja daje vrednost broja π (3,1415927)

PRIMER: `10 PRINT PI'PI/2'LN PI`

TRIGONOMETRIJSKE FUNKCIJE

Trigonometrijske funkcije kojima Spektrum raspolaže su:
SIN x, funkcija nalazi sinus ugla x datog u radijanima.
1 radijan = 180/PI stepeni odnosno 360° = 2*PI radijana

PRIMER:

```
10 FOR n=0 TO 255
20 PLOT n,88+80*SIN (n*PI/128)
30 NEXT n
```

Na ekranu ća se poja. i funkcija sin x kada se x manja od 0 do 2*PI radijana.

COS x, kosinus ugle x datog u radijanima

PRIMER zameniti liniju 20 u donjem primeru sa

```
20 PLOT n,88+80*COS (n*PI/128)
```

TAN x, tangens ugla x datog u radijanima

PRIMER:
```
10 PRINT TAN 0'TAN (PI/3)'TAN
(2*PI/3)'TAN (PI/2)
```

ASN x, arkussinus broja x (koji mora biti u opsegu od –1 do 1)

PRIMER:
```
10 FOR n=0 TO 255
20 PLOT n,88+175/PI*ASN (n/128
-1)
30 NEXT n
```

ACS x, arkuskosinus broja x (koji mora biti u opsegu od –1 do 1)

PRIMER:
```
10 FOR n=0 TO 255
20 PLOT n,88+175/PI*A    (n/128
-1)
30 NEXT n
```

ATN x, arkustangens broja x. Vrednost ove funkcije je uvek između –PI/2 i PI/2.

PRIMER:
```
10 PRINT ATN -123456'ATN 0'ATN
123456
```

Argument ovih funkcija (x) može biti zadat i u obliku promenljive ili izraza.

OPERACIJE POREĐENJA

Spektrum omogućava sledeće operacije poređenja:

= jednako
< manje
> veće
<= manje ili jednako
>= veće ili jednako
<> različito

Mogu se porediti brojevi ili stringovi, dati neposredno ili kao promenljive ili izrazi.

Ako je ishod poređenja tačan, rezultat operacije će biti 1, a ako nije, rezultat će biti 0.

PRIMER:

naredba	rezultat
PRINT 5>3	**1**
PRINT 5<3	**0**

Stringovi se porede tako što se porede kodovi prvih keraktera. Ako su jedneki, porede se kodovi drugih karektera u tim stringovime, eko su i oni jednaki, porede se treći i teko redom.

PRIMER:

neredbe	**rezultat**
PRINT "aab">"ab"	**1**
PRINT "ah"<"lh"	**0**

PRIMER: ispisuju se rezulteti svih vrsta poredenja dva broja

```
10 INPUT "x*";x,"y=";y
20 PRINT x=y'x<y'x>y'x<=y'x>=y
'x<>y
```

PRIMER Neposrednom upotrebom poređenja se omogućava pravljenje krećih i bržih programa

```
10 FOR n=0 TO 25
20 PRINT AT (n>15)*(n-15),n;"+
"
30 NEXT n
```

Poredenja se vrlo često koriste keo uslov u naredbi IF THEN.

LOGIČKE OPERACIJE

Logičke operacije se vrše ned jednim ili dva argumenta, koji mogu biti dati u obliku promenljivih ili izraza.

NOT x, Operacija negecije obavlja se nad jednim argumentom. Ako je on Ø rezultat operacije je 1, a ako je argument različit od Ø rezultat je Ø.

PRIMER:

```
10 INPUT a
20 PRINT NOT a: GO TO 10
```

x **AND** y, Operacija logičkog množenja (logičko I). Prvi argument je broj ili string, a drugi je uvek broj.
Prvi argument je broj:

x AND y = x ako je y<>0
x AND y = 0 ako je y = 0

Prvi argument je string

x$ AND y = x$ ako je y<>0
x$ AND y = "" (prazan string) ako je y=0

PRIMER

```
10 FOR b=-5 TO 5
20 PRINT 299792 AND b
30 NEXT b
```

PRIMER

```
10 FOR b=-5 TO 5
20 PRINT "MNOZENJE" AND b
30 NEXT b
```

x **OR** y. Operacija logičkog sabiranja (logičko ILI). Izvršava se nad dva brojna argumenta

x OR y = 1 ako je y<>0
x OR y = 0 ako je y = 0

PRIMER

```
10 FOR b=-5 TO 5
20 PRINT 299792 OR b
30 NEXT b
```

PRIORITETI FUNKCIJA I OPERACIJA

Funkcije i operacije se obavljaju prema prioritetu, prvo one sa višim, a zatim one sa nižim prioritetom.

Prioritet operacije

najviši	1	formiranje indeksa i podstringova
	2	sve funkcije sem NOT i predznak minus
	3	stepenovanje
	4	predznak minus
	5	množenje i deljenje
	6	sabiranje i oduzimanje
	7	relacije (=, >, <, >=, <=, <>)
	8	NOT
	9	AND
najniži	10	OR

RND

RND se koristi za dobijanje slučajnog broja koji se nalazi u opsegu od 0 do 1 (1 nije obuhvaćen).

Primena slučajnih brojeva je široka, od raznih naučnih disciplina, do igara.

PRIMER

```
10 FOR n=1 TO 10
20 PRINT RND
30 NEXT n
```

U ovom primeru se ispisuje deset slučajnih brojeva.

Pomoću **RND** se ne dobijaju pravi slučajni brojevi već tzv. pseudoslučajni brojevi. Oni čine niz od 65536 različitih brojeva koji počinju da se ponavljaju posle 65536 brojeva. Izračunavaju se na osnovu izraza

(75 * (n+1) – 1) / 65536

n je vrednost prethodnog slučajnog broja.
Početna vrednost n je vrednost promenljive SEED koja se postavlja naredbom **RANDOMIZE.**
PRIMER: Na sledeći način se može dobiti 5 slučajnih celih brojeva u opsegu od 1 do 36, što odgovara izvlačenju brojeva u igri LOTO

```
10 FOR n=1 TO 5
20 PRINT 1+INT (36*RND)
30 NEXT n
```

RANDOMIZE n

Broj n koji se navede iza naredbe **RANDOMIZE** se koristi kada se sledeći put pomoću **RND** izračunava slučajni broj. n može da bude u opsegu od 0 do 65535. Može biti zadat i u obliku promenljive ili izraza, a ako nije ceo broj, zaokružuje se.

Ova naredba svoju ulogu ostvaruje tako što sistemskoj promenljivoj SEED dodeljuje vrednost n. Ako je n nula ili se iza **RANDOMIZE** ne stavlja nikakav broj, promanljivoj SEED će biti dodeljena vrednost sistemska promenljive FRAMES, a ona je jednaka broju poluslika koje su prikazane na akranu od trenutka uključenja računara.

BIN

BIN omogućava unošenje brojeva u binarnom obliku.
PRIMER:

```
10 LET a=BIN10110
20 PRINT a
```

U liniji 10 promenljivoj a dodeljuje se vrednost 22 decimalno (10110 binarno), a naredbom u liniji 20 ta vrednost će biti ispisana na ekranu.

Uneti binarni broj može imati maksimalno 16 bita. To znači da na ovaj način mogu da se unose vrednosti od 0 do 65535 (samo celi brojevi).

DEF FN

U programu sa neki izraz može ponavljati na više mesta. U tom slučaju je pogodno definisati funkciju, kojoj se dodeljuje vrednost tog izraza. Tada je dovoljno da se umesto izraza navedu ima funkcije i njeni argumenti.

Naredba **DEF FN** služi za definisanje novih funkcija. Njen opšti oblik je

DEF FN $a(b_1,....,b_n)$ = izraz

a je ime funkcija, a b_1 do b_n su imena promenljivih koje su argumenti funkcije i nalaze se u zagradama. I kada nema argumenata, zagrade moraju da postoje. Sa desne strane znaka jednakosti navodi se željeni izraz. Ime funkcije može biti samo jedno slovo. Između velikog i malog slova ne se pravi se razlika. Dozvoljeni su i karakteri boje i karakteri praznog prostora. Imena argumenata mogu takođe imati samo jedno slovo. Broj argumenata nije ograničen.

Funkcija i argumenti mogu biti brojne i string promenljive.

PRIMER: Definisanje brojne funkcije čiji je jedan argument brojna, a drugi string promenljiva.
DEF FN a(x,y$)=x*LEN y$

PRIMER: Definisanje string funkcije, koja ima dva brojna i jedan string argument

```
10 DEF FN b$(z,w,j$)=CHR$ z+ST
R$ w+j$
```

Naredba **DEF FN** se može postaviti bilo gde u programu. Ona se izvršava tek kada se izvršava naredba **FN**. Poželjno je da se kod dužih programa postavi bliže početku programa radi bržeg izvršavanja.

FN

Nalaženje vrednosti funkcije definisane naredbom **DEF FN** se ostvaruje naredbom **FN**. Iza nje sledi ime funkcije i argumenti, koji se nalaze u zagradama. Argumenti mogu biti dati kao brojevi, odnosno stringovi, ili kao brojne ili string promenljive ili izrazi.

PRIMER:
```
  10 DEF FN a(x,y$)=x*LEN y$
  20 DEF FN b$(w,j$.z)=STR$ w+j$
+CHR$ z
  30 PRINT FN a(10."abc")
  40 PRINT FN b$(RND." mikro ",6
5)
```
PRIMER:
```
  10 LET b=3
  20 LET c=4
  30 DEF FN a(b,c)=b*c
  40 DEF FN g()=b*c
  50 PRINT FN a(5.6)
  60 PRINT FN g()
```
U liniji 30 je zadata funkcija e čiji su argumenti b i c, a u liniji 40 funkcija g bez argumeneta. Obe funkcije su potpuno iste, eli se pri izračunavenju prve koriste vrednosti promenljivih b i c koje su zedate kao argumenti (5 i 6), a prilikom izračunavanje druge se koriste vrednosti promenljivih koje su definisane u programu (3 i 4).

RAD SA STRINGOVIMA

Sabiranje stringova (+)

Stringovi se mogu sabirati tako što se između njih postavi operator +.

PRIMER:

naredba	rezultat
PRINT "ab"+"c2E"	**abc2E**

Umesto stringova se mogu koristiti string promenljive i izrazi. Važno je uočiti da u slučaju zemene mesta stringove rezultat nije isti.

PRIMER:
```
  10 LET a$="BEO": LET b$="GRAD"
  20 PRINT a$+b$.b$+a$
```
Podstringovi

Podstring je deo stringa Dobija se tako što se izostave neki početni i/ili neki krajnji karakteri stringa na sledeći način

string (a TO b)

a ja broj mesta u stringu onog karaktere koji se uzima kao prvi karaktar podstringa, a b poslednjeg karaktera podstringa. Broj mesta prvog karaktera u stringu je 1.

PRIMER: naredba **PRINT "abc123" (3 TO 5)** rezultat **c12**

String c12 je podstring stringa abc123.

Brojevi prvog i poslednjeg karaktera podstringa mogu biti dati kao promenljive ili izrazi. Ako vrednosti nisu cele, biće zaokružene. U slučaju da su zadati brojevi manji od 1 ili veći od broja karaktera u stringu, javiće se greška sa izveštajem **3 Subscript wrong** (pogrešan indeks).

PRIMER:

```
10 LET a$="abcdef"
20 PRINT "a$",a$
30 FOR n=1 TO 6: FOR m=n TO 6
40 PRINT "a$(";n;" TO ";m;")",
a$(n TO m)
50 NEXT m: NEXT n
```

Ako se ne navede broj mesta prvog karaktera podstirnga, smatraće se da je to prvi karakter stringa. Isto tako, ako se ne navade broj masta posladnjeg karaktara podstringa, smatraće se da je to poslednji karakter stringa.

PRIMER:

```
10 PRINT "qwerty"( TO 3)
20 PRINT "qwerty"(2 TO )
```

CHR$ n

Svakom Spektrumovom karakteru (slovo, broj, znak, kontrolni karakter, ima naredba) je pridodat jedan broj koji se naziva kôd karaktera (dati su u dodatku).

Naredbom **CHR$** se dobija karaktar čiji je kôd zadati broj n.

PRIMER: kôd slova A je 65 pa je rezultat naredbe **PRINT CHR$ 65** slovo A.

Broj može biti zadat i kao promenljiva ili brojni izraz, a ako nija cao broj, biće zaokružen.

PRIMER: Pomoću sledaćeg programa sa prikazuju karakteri čiji su kodovi između 32 i 255.

```
10 FOR n=32 TO 255
20 PRINT CHR$ n;
30 NEXT n
```

Kodovi od 6 do 23 pripadaju kontrolnim karakterima. Oni se mogu koristiti i pomoću naredbe **CHR$** mada se isto može postići i na pogodnije načine.

PRIMER: **CHR$ 8** ća mesto na kome se ispisuje pomeriti u levo.

```
10 PRINT AT 5,5;"↑";CHR$ 8; OV
ER 1;"_"; OVER 1
```

Znaci ↑ i _ se ispisuju jedan preko drugog.
PRIMER: **CHR$ 16** menja boju zapisa na ekranu **(INK)**.

```
10 PRINT "NOVI";CHR$ 16;CHR$ 3
;" SAD"
```

Boja zapisa se menja u purpurnu (broj 3).

CODE n$

Ovom naredbom se nalazi brojna vrednost – kôd prvog karaktera naznačenog stringa. Ako je string prazan, tj. bez ijednog karaktera, kôd koji se dobija je 0.

PRIMER:

naredba	rezultat
PRINT CODE "A"	**65**
PRINT CODE ""	**0**

String može biti dat u obliku promenljive ili izraza. Može se reći da ova naredba ima inverzan afekat od naradbe

CHR$. Dok se pomoću **CHR$** prevodi brojna vrednost u karakter, CODE prevodi karakter u brojnu vradnost.

LEN x$

Pomoću **LEN** se dobija broj karaktera u stringu koji je naveden iza naredbe.

PRIMER:

naredba	rezultat
PRINT LEN "abc"	3

String može biti dat u obliku promenljiva ili izraza.

PRIMER:

```
      10 INPUT "Upisite vase puno im
    e i prezime",a$
      20 FOR n=1 TO LEN a$
      30 PRINT a$( TO n)
      40 NEXT n
```

STR$ n

Naredba **STR$** prevodi broj naveden iza nja u string čiji su karakteri cifre toga broja.

PRIMER	naredba	rezultat
	PRINT STR$ 67	**67**
	ovo je broj	ovo je string

PRIMER:

```
      10 LET a$=STR$ 12
      20 PRINT a$
      30 LET b$=STR$ (104+2.31)
      40 PRINT b$
      50 PRINT a$+b$
```

VAL n$

Naredba **VAL** prevodi string (n$), čiji su karaktari brojevi, u odgovarajući brojnu vrednost

PRIMER:	naredba	razultat
	PRINT VAL "2+3.1"	**5.1**

String može biti dat i u obliku promenljive ili izraza.

PRIMER:

```
      10 LET a$="132313"
      20 PRINT VAL a$
```

**VAL$** x$

Naredbom **VAL$** se prevodi string u string. U naredbi **VAL** argument se, unutar navodnika, tretira kao broj, a u **VAL$** kao string. što nameće upotrebu dodatnog para navodnika za označavanje početka i kraja stringa.

PRIMER:	naredba	rezultat
	PRINT VAL$	**2+3.1**
	"""2+3.1"""	

String argument se može dati kao promenljiva ili izraz.

PRIMER:
```
10 LET a$="xyz78"
20 PRINT VAL$ "a$"
```

USR string

Spektrum pruža mogućnost da korisnik sam definiša 21 karektar. Svaki od njih je pridružan jednom tastaru od A do U i svakome ja dodeljano osem bajta u memoriji pomoću kojih je on definisen. Ti bajti čine oblast u memoriji koja je rezervisana za grefiku (karektere) dafinisanu korisnikom (engl. user dafined graphic – UDG). Pomoću **USR**, iza koga ja i pod navodnicima jedno slovo od A do U, dobija se adresa prvog od osam bajta koji pripadaju karakteru dafinisanom uz tastar sa tim slovom.

PRIMER:	naredba	rezultat
	PRINT USR "a"	**32600** (18K Spaktrum)
		65368 (48K Spektrum)

Dobijeni razultat je adresa prvog od osam bajta kojim se definiše karaktar uz taster **a**. Adresa drugog bajta je **USR "a"+1**, trećeg **USR "a" +2** i poslednjeg **USR "a" +7**. Adresa **USR "a" +8** je jedneka adresi **USR "b"**.

Ne pravi se razlika izmedu velikih i malih slove. String iza **USR** se može dati i u obliku promenljive ili izraza, ali može sadržati samo jedno slovo.

NAČIN DEFINISANJA KARAKTERA

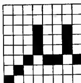
Definisaćemo karakter u obliku grčkog slova µ koje će se javljati pritiskom na tastar M.

U polju od 8x8 tačaka odrediće sa one koje trabe da budu zatamnjene.

Bit koji odgovare tečki koja je zatemnjena treba da bude 1. Prvi bejt odgovara gornjem redu tačaka, a poslednji (bejt 8) donjem redu tačaka.

Na odgovarajuće adrese u memoriji se upisuju potrebni bajti U ovom slučaju oni će biti navedeni u binarnom obliku

```
10 FOR n=USR "m" TO USR "m"+7
20 READ a
30 POKE n,a
40 NEXT n
100 DATA BIN 00000000
110 DATA BIN 00000000
120 DATA BIN 00010010
130 DATA BIN 00010010
140 DATA BIN 00010010
150 DATA BIN 00101101
160 DATA BIN 01000000
170 DATA BIN 10000000
```


Slika 3-2 Karakter slova µ

Izvršavanjem ovog programa karakter je definisan. Na ekranu će biti prikazan pritiskom tastara M dok je pokazivač u G obliku.

PEEK n

Pomoću **PEEK** se može očitati sadržaj memorijske lokacije čija je adresa n. Zadata adresa (n) mora biti u opsegu od 0 do 65535. Ako nije navaden cao broj, vrši se zaokruživanja. Adresa se može dati i u obliku izraza ili kao promanljiva.

Pošto se na svakoj memorijskoj lokaciji nalazi jedan bajt, njan sadržaj ća uvek biti broj od 0 do 255.

PRIMER `10 PRINT PEEK 23756`

Pri izvršavanju ove linije ispisuje se broj 10, što je sadržaj memorijske lokacije sa adresom 23756. U njoj se nalazi broj upravo te programske linije (bajta manje težina).

POKE n,m

Naredbom **POKE** se u memorijsku lokaciju sa adresom n upisuje broj m. Adresa (n) može biti u opsegu od 0 do 65535. Vrednost koja se upisuje (m) mora biti između -255 i 255. Kada je dat negativni broj, upisuje se zbir između 256 i tog broja (npr. za m= −2 upisace se 254).

I za adresu i za sadržaj koji se upisuja važi da će biti zaokruženi ako nisu celi brojavi i da se mogu dati kao promenljive ili broj-ni izrazi.

PRIMER: `5 POKE 23756,0`

Izvršavajući ovu liniju izmenićemo njan broj, jer se na adresi 23756 nalazi upisan broj te linije.

USR broj

Pomoću naredbe **USR**, iza koja je naveden broj, otpočinja izvršavanja mašinskog programa od memorijske lokacije čija je adrasa jednaka tom broju.

PRIMER:
```
10 FOR n=1 TO 700
20 PRINT "+";: NEXT n
```

U toku izvršavanja ovog programa ispisuje se po ekranu znak + Program treba prekinuti (**BREAK**) i zatim dati direktnu naredbu **RANDOMIZE USR 7775**. Tada ća se izvršiti mašinski program u ROM-u, čija je prva naredba na adresi 7775. On ostvaruja naredbu **CONTINUE**, pa će se prakinuti program nastaviti (nastaviće se ispisivanje znaka +).

PRIMER: Mikroprocesor posle uključivanja počinja da izvršava mašinski program koji počinje na adresi 0.

Izvršavanjem narede **PRINT USR 0** dobićamo isti afakat kao posle uključivanja.

Broj koji se navodi mora biti između 0 i 65535. Može biti dat u obliku promenljive ili izraza, a ako nije cao broj, zaokružuje se na najbližu celobrojnu vrednost.

USR n se ne može upotrebljavati samostalno, pa sa koristí najčešće pomoću **PRINT USR** n i **RANDOMIZE USR** n.

Kada se za izvršavanje mašinskog programe upotrebljava **USR**, nakon njegovog završetka se dobija sadržaj ragistar para BC (poglavlje 5).

Ovo ja jadna od najsnažnijih naredbi bejzika jer omogućava korišćenje valikih mogućnosti mašinskih programa.

CLEAR n

Ova naredba briše sva promenljiva oslobadajući prostor u memoriji. Ima i efekat **RESTORE** naredbe, briše sadržaj ekrana kao **CLS**, a **PLOT** poziciju postavlja u donji levi ugao ekrana. Takođe briša i **GO SUB** stek iz memorije.

Ako se iza **CLEAR** navede broj n, sistemskoj promenljivoj RAMTOP će se dodeliti ta vrednost. Broj n se može dati u obliku promenljive ili izraza. U slučaju da broj n nije ceo, zaokružiće se na najbližu celobrojnu vrednost.

IN x

Ovom naredbom se čita podatak sa ulazne periferne jedinice čija je adresa x. Slična je naradbi **PEEK** x kojom se čita sadržaj memorijske lokacije čije je adrasa x. x može biti broj od 0 do 65535. Moguće je zadati ga i u obliku promenljive ili brojnog izraza, a ako nije cao, zaokružuje sa na najbliži celobrojnu vrednost. Pošto je dobijani podatak jedan bajt, rezultat ove naradbe može imati vrednost od 0 do 255.

PRIMER:
```
10 PRINT AT 5,5;IN 254: GO TO
10
```

Ovo je beskonačna petlja u kojoj se neprestano čita i ispisuje podatak sa ulazne jedinice čija je adresa 254. Biti B0 do B4 učitanog bajta nose informaciju o stanju tastatura, a bit B6 o stanju na ulazu sa kasetofona (EAR). Pritiskanjam tastera, dok se ovaj program izvršava, menjaće se broj koji se prikazuje na ekranu.

OUT x,y

Ova naredba upisuje u izlaznu jedinicu sa adrasom x, podatak y. Slična je naredbi **POKE** x,y kojom se u memorijsku lokaciju sa adesom x upisuje podatak y. Broj x (adresa) može imati vrednost u opsegu od 0 do 65535, a broj y (podatak) u opsagu od 0 do 255 (jedan bajt). Oba broja mogu biti data i u obliku promenljivih i izraza, a u slučaju da nisu cali, zaokružuju sa na najbliži ceo broj.

U sledećim primarima će se koristiti izlazna jedinica čija je adresa 254. Njani biti B0, B1 i B2 određuju boju obodnog dela slike, bit B3 određuje stanje na izlazu za kasetofon (MIC) a bit B4 utiče na rad zvučnika.

PRIMER
```
10 FOR n=0 TO 7
20 OUT 254,n
30 PAUSE 2
```

```
40 NEXT n
50 GO TO 10
```

Ovim programom se u intervalima vremena odradnom naradbom **PAUSE** menja boja obodnog dela ekrana tako što se na izlaznu jedinicu sa adresom 254 šalju brojevi od 0 do 7.

PRIMER:
```
10 OUT 254,16
20 OUT 254,0
30 GO TO 10
```

Pomoću ovakvog programa sa manja stanje na zvučniku ma-njajući bit B4 na izlaznoj jedinici sa adresom 254. Od brzine izvrša-vanja programa zavisi visina tona.

BOJE NA SPEKTRUMU

Spektrum može da na ekranu prikaže ukupno 8 boja, uključujući crnu i belu. Svaka od boja je predstavljena odgovarajućim brojem iznad tastera sa brojevima su naznačene boje.

0 crna	4 zelena
1 plava	5 svetloplava
2 crvena	6 žuta
3 magente (purpurna)	7 bela

Slika na ekranu sa sastoji od 24 reda i 32 kolone što čini ukup-no 768 elementarnih polja. U jadno polje se može upisati jedan od znakova, tj. jedan karakter.

Svaki karaktar se sastoji od 8×8 tačaka (videti 3–6). Tačke koje se ispisuju formiraju zapis **(INK)** koji je obično (ako nije druga-čije naznačeno) crna boje. Ostale tačke se ne ispisuju ostavljajući osnovu **(PAPER)** u beloj boji.

I zapis i osnova mogu imati boju koju zahteva korisnik. Unutar jednog od 768 polja mogu postojati samo dve boje. Jedna odgovara boji zapisa, a druga boji osnove.

INK n

Naredba za zadavanje boje zapisa

PRIMER: `10 PRINT "A"; INK 4;"B"`

Sa **INK 4** je određena zelena boja slova B. Naredba takođe omogućuje i zadavanje boje zapisa na celom ekranu (sa izuzetkom editorskih linija).

PRIMER: `10 INK 5: CIRCLE 127,87,87`

Svaki naredni zapis na ekranu će biti svetloplave **(INK5)** boje, sve do izvršenja naredbe koja će postaviti novu boju.

Dozvoljene vrednosti za broj n koji određuje boju su od 0 do 9. Sa vrednostima od 0 do 7 se dobijaju navedene boje. Za vrednost 8 boja zapisa će biti ona koja je ranije bila definisana.

PRIMER:
```
10 PRINT AT 10,10; INK 2;"A";
INK 5;"B"
20 PAUSE 0
30 PRINT AT 10,10; INK 8;"C";"
D"
```

U ovom programu se linijom 10 ispisuje slovo A crvene i slovo B svetloplave boje. U liniji 20 se čeka pritisak na bilo koji taster. Nakon toga linijom 30 se ispisuju slova C i D sa bojama koje su bila ranije definisane za polja u kojima se ispisuju.

Vrednost 9 navadana iza naredbe **INK** omogućuje kontrastno ispisivanje. Na osnovama tamnih boja (0–3) zapis će biti beo (7), a na osnovama svetlih boja (4–7) biće crn (0).

PRIMER:
```
10 FOR n=0 TO 7: PRINT  INK 9;
PAPER n;"A": NEXT n
```

Vradnost boje može biti zadata u obliku promenljive ili izraza. Ako vrednost nije cela, zaokružuje se na najbližu celobrojnu. U slučaju nadozvoljenih vradnosti pojaviće se izveštaj o grašci **K Invalid color** (nedozvoljena boja).

PAPER n

Naredba za zadavanje boje osnove. Važi sve izneseno za naredbu boje zapisa **(INK** n) samo što se ovde primenjuje za osnovu

To znači da je u sva četiri prethodna primera potrebno zameniti naredbu **INK** naredbom **PAPER**, a u četvrtom primeru umesto naredbe **PAPER** treba staviti naredbu **INK**.

Izuzetak je zadavanje boje osnove celog akrana. Tada je iza naredbe boje osnove potrebno upotrabiti naredbu brisanja sadržaja ekrana **CLS**. To znači da drugi primer za prethodnu naredbu glasi:

```
10 PAPER 5: CLS : CIRCLE 127,B
7,B7
```

BORDER n

Obodnom delu slike se može dati jadna od osam boja. Potrebno je iza naredbe **BORDER** navesti broj koji odgovara željenoj boji. Taj broj može biti zadat i u obliku promenljive ili izraza, a ako nije ceo, zaokružuje se

PRIMER:
```
10 FOR n=7 TO 0 STEP -1: PAUSE
20: BORDER n: NEXT n
20 GO TO 10
```

Obodni deo slike redom dobija sve boje.

PRIMER:
```
10 BORDER 0: BORDER 1: BORDER
2: BORDER 3: BORDER 4: BORDER 5:
BORDER 6: BORDER 7: PAUSE 1: GO
TO 10
```

Ovaj program boji obodni deo slika svim bojama odjednom.

BRIGHT n

Naredba za zadavanje osvetljenosti.

PRIMER:
```
10 PRINT "A"; BRIGHT 1;"B"
```

Naredbom **BRIGHT 1** polje karaktera B je ispisano povačanom osvetljenošću.

Vrednost 1 navedana iza **BRIGHT** definiše povaćanu osvetljenost, a vrednost nula definiše normalnu osvetljenost. Dozvoljena je još vrednost 8, koja (isto kao u slučaju naredbi **INK 8** i **PAPER 8**) omogućava da novo ispisivanje bude isto kao prathodno.

Naredba osvetljenosti se može upotrebiti i za postavljenje povećane osvetljenosti celog ekrana (bez okvira).

PRIMER:
```
10 BRIGHT 1: PRINT "Osvetljeno
st"
```

Izvršavanjem primera poruka "Osvetljenost" će se ispisati u povećenoj osvetljenosti. Nakon toga pritiskom na taster **ENTER**, ceo ekren će preći u povećenu osvetljenost, što ukazuje na isti mehenizam definisanja celog ekrana, kao u slučeju naredbe **PAPER**. Potrebno je nakon naredbe **BRIGHT 1** upotrebiti **CLS** naredbu. Vraćanje u normelnu osvetljenost celog ekrana može se postići naredbom **BRIGHT 0** uz dva pritiska ne tester **ENTER**.

Vrednosti osvetljenosti se mogu dati i izrazima i promenljivema.

FLASH n

Ovom neredbom se omogućave da pojedini karakteri, ili cela slike trepti. Ako je n = 1 jevlja se treptanje, tako što osnova i zapis u kratkim vremenskim intervalima medusobno zamenjuju boje. Ako je n = 0, treptanja neme, e ako je n = 8, zadržava se prethodno postavljeno stanje za karekter na tom mestu.

PRIMER:
```
10 PRINT "aaaaaaaaaa"
20 PRINT AT 0,2; FLASH 1;"FFFF
"
30 PAUSE 0
40 PRINT AT 0,4; FLASH 8;"8888
"
```

Ispisuju se slova a koja ne trpte, a zetim slova F koja trepte. Posle pritiske na bilo koji tester ispisuju se osmice, dve trepte, a dve ne trepte, jer je tako ranije definisano za meste na kojima su ispisane. Naredba **FLASH** se primenjuje ne isti način kao i naredba **BRIGHT**.

INVERSE n

Ovom naredbom se mogu medusobno zameniti boje zapisa i osnove. Ako je broj naveden ize **INVERSE** jednak jedinici, boje zapisa i osnove se zemenjuju, a ako je iza **INVERSE** navedena nula, boje će biti onakve kekve su ranije definisane.

PRIMER.
```
10 PRINT  PAPER 6; INK 3;"AAAA
   AAA"'' INVERSE 1;"BBBBBBB"
```

Slova A će biti ispisana purpurnom (megente) bojom ne žutoj osnovi, a slova B, za koja je zadata naradba **INVERSE 1,** biće ispisana žutom bojom na purpurnoj osnovi.

Boje osnove i zapise mogu biti međusobno zamenjene i na celom ekranu izvršavanjem naredbe **INVERSE 1,** direktno ili programski. Pomoću naredbe **INVERSE 0** vraća se u prethodno stanje.

Jedine dozvoljene vrednosti za n su 0 i 1. Mogu se dati i u obliku promenljivih ili izraza, a ako nisu celi brojevi, biće zaokruženi ne najbliži ceo.

OVER n

Naredbom **OVER** se, ako je ize nje naveden broj 1, omogućave ispisivanje novog karaktera preko starog, ali teko da sa zepis gubi u tačkama u kojima zapis postoji i u starom i u novom karekteru. Ako je navedene nule, prilikom ispisivanja novog karaktera, stari, koji se nelazio na tom mestu, potpuno se briše.

PRIMER:
```
10 PRINT AT 10,15;"A"
20 PAUSE 0
30 PRINT AT 10,15; OVER 1;"O"
```

Nakon ispisivanja slova A, pritiskom na taster će se izvršiti naredbe u liniji 30 pomoću kojih će se slovo 0 ispisati preko A. Zapis će postojati samo u onim tačkama u kojima zapis postoji kod semo jednog (ali ne i oba) karaktera.

Prilikom ispisivanja novog karaktere korisno je upotrebiti kontrolni karekter **CHR$ 8**, pomoću koga se pokazivač pomera u levo.

PRIMER:
```
10 OVER 1
20 PRINT "_";CHR$ 8;"I"
30 OVER 0
```

DIREKTNO ZADAVANJE BOJE

Boja, osvetljenost i treperenje se mogu kontrolisati i direktno preko tastature, a ne samo naredbama. Željene promene se dobija-

	SHIFT	1	2	3	4	5	6	7	8	9	Ø
K,L ili C	SYMBOL	!	@	#	$	%	'	&	(	)	_
	CAPS	EDIT	CAPS LOCK	TRUE VIDEO	INVERSE VIDEO	⇦	⇩	⇧	⇨	RAD SA GRAFI KOM	DELETE
G										PREKID RADA SA GRAFI KOM	DELETE
	SYMBOL ILI CAPS										DELETE
E		plava osnove	crvena osnove	magenta osnova	zelena osnova	svetlo plava osnova	žuta osnova	bela osnova	normelne osvetlje nost	poječan osvetljaj	crne osnova
	SYMBOL	DEF FN	FN	LINE	OPEN #	CLOSE #	MOVE	ERASF	POINT	CAT	FORMAT
	CAPS	plavi zapis	crvani zapis	magenta zapis	zeleni zapis	svetlo plavi zapis	žuti zapis	beli zapis	isključ treptanja	uključ treptanje	crni zapis

Tabela 3–1 Upotreba gornjeg reda tastera

ju pritiskom na jedan od tastera gornjeg reda kada je pokazivač E. Pri tom ja u nekim slučajevima potrebno držati pritisnut i **CAPS SHIFT**.

U sledećoj tabeli su prikazani načini direktne kontrole boja i druge funkcije gornjeg reda tastera.

PLOT x,y

Ovom naradbom se na mestu sa koordinatama x i y crta tačka. Koordinatni početak je donji levi ugao gornjeg dela ekrena.

PRIMER:
```
10 PLOT 0,0
```

Na ovaj način se dobija tečka u koordinatnom početku.

Ukupan broj tačaka gornjeg dela akrana, na kome može da se crta pomoću naradba **PLOT**, je 256 × 176 × osa (prvi broj) je orijentisana na dešno, a y osa (drugi broj) na gore. Koordinate gornjeg dasnog ugla su x = 255, y = 175, što sa moža proveriti naredbom:

```
10 PLOT 255,175
```

Argumenti naredba **PLOT**, tj. koordinate, mogu biti data i kao promenljiva ili izrazi.

PRIMER.
```
10 FOR n=0 TO 100
20 PLOT n,n*n/100
30 NEXT n
```

Pomoću ovog programa sa crta deo funkcije $n^2/100$

Iza naredbe **PLOT**, ispred koordinata, može se navesti naredba **INVERSE** 1. Na ovaj način sa nacrtane tačke mogu brisati.

PRIMER:
```
10 PLOT 100,100: PAUSE 0
20 PLOT  INVERSE 1;100,100
30 PAUSE 0
```

Prvom naredbom **PLOT** se na koordinatama 100,100, crta tačka, a posle pritiskanja bilo kog tastera, ta tačka sa briša pomoću naradba **PLOT** u liniji 20.

iza naredbe **PLOT** je moguće upotrebiti i naredbu **OVER 1** pomoću koje se na nekom mestu može nacrtati tačka, ako tu nema zapise, a ako tačka već postoji, biće izbrisana

PRIMER
```
10 FOR n=0 TO 200
20 PLOT  OVER 1;n,n/2
30 NEXT n
```

Ovim programom će biti nacrtana linija. Ako se ponovo izvrši direktnom naredbom **GO TO 10**, ova linija će biti izbrisane

DRAW x ,y,l

Pomoću naredbe **DRAW** iza koje su navedena dva broja (x i y) crta se prave linija između poslednje nacrtane tačke i tačke koja u odnosu na nju ima koordinete x i y. Koordinata x je orijentisana na desno, a koordinata y na gore

PRIMER
```
10 PLOT 100,20: DRAW 150,50
20 PAUSE 0
30 DRAW -10,100: DRAW -140,-60
```

U ovom primeru prvo se crta početna tačka na koordinetama x = 100 i y = 20, a zetim se povlači linije do tačke sa koordinatama x = 250 i y = 70 (150, 50 u odnosu na početnu tačku) Nakon pritiska bilo kog testera biće nacrtane još dve linije Koordinate su zadete u odnosu na poslednje nacrtene tačke

U slučaju da na ekranu nije nacrtena nijedna tačka, početna tačka je koordinatni početak (donji levi ugao gornjeg dela ekrana)

PRIMER Izvršiti program iz gornjeg primera kada je u liniji 10 izostavljene naredba **PLOT** 100,20

Vrednosti koordinata se mogu zadati kao promenljive ili izrazi a ako nisu celi brojevi, zaokružuju se na cele

PRIMER
```
10 FOR n=0 TO 255 STEP 3
20 PLOT 0,0: DRAW n,175
30 NEXT n
40 FOR n=0 TO -255 STEP -3
```

```
50 PLOT 255,175: DRAW n,-175
60 NEXT n
```

Naredbom **DRAW** iza koje su navedena tri broja crtaju se kružni lukovi. Luk počinje u tački koja je poslednja nacrtana, a završava se u onoj koja je koordinatama x i y data u odnosu na početnu. Treći navedeni broj, I, predstavlja, ugao kružnog isečka koji odgovara tom luku i dat je u radijanima. Ako je I pozitivan broj, luk će se iscrtati u smeru obrnutom od smera kazaljke na satu, a ako je negativan, biće iscrtan u smaru kretanja kazaljke na satu.

PRIMER.

```
10 PLOT 10,30: DRAW 100,80,PI:
DRAW 100.50,PI
20 DRAW -150,-60,-PI: DRAW -40
,30,PI/2
```

Naredbama u liniji **10** se jedan za drugim crtaju dva luka (polukruga zbog zadatih PI radijana) u smaru suprotnom od kretanja kazaljke na satu, zatim naredbama u liniji **20** jedan luk u smeru kretanja kazaljke na satu i na kraju četvrtina kruga (PI/2) u smeru suprotnom od kretanja kazaljke na satu.

Naradba **DRAW**, kao i naredba **PLOT**, dozvoljava upotrebu naredbi **INVERSE 1** i **OVER 1**. Efekat njihove primane je isti kao kod naradbe **PLOT**.

PRIMER.

```
10 PLOT 0,50: DRAW  OVER 1;150
,80,PI: GO TO 10
```

CIRCLE x,y,r

Ovom naradbom sa crta kružnica. x i y su koordinate centra kružnice, a r je njen poluprečnik. Koordinatni sistam je isti kao kod naredbe **PLOT**. Argumanti (x, y i r) mogu biti dati i kao promenljiva ili izrazi.

PRIMER:

```
10 FOR r=20 TO 80 STEP 8
20 CIRCLE r,85,r
30 CIRCLE 254-r,85,r
40 NEXT r
```

I u ovoj naredbi se mogu upotrebiti naredbe **INVERSE 1** i **OVER 1**.

POINT (x,y)

Rezultat ove funkcije je 1 ako se na koordinatama x, y nalazi tačka boje zapisa **(INK)**. Ukoliko to nije slučaj, tj. tačka je boje osnove, rezultat će biti 0.

Koordinatni sistem je isti kao kod naredbe **PLOT**: x je broj od 0 do 255, a y od 0 do 175. Argumenti (x i y) mogu biti i promenljive i izrazi.

PRIMER:
```
10 PLOT 60,160
20 PRINT POINT (60,160),POINT
(61,160)
```

Rezultat prve naredbe **POINT** je 1 jer na tim koordinatama postoji tačka, a rezultat druge naredbe **POINT** je 0 jer tu nema tačke.

ATTR (x,y)

Pomoću funkcije **ATTR** se dobije vrednost atributa karaktera koji se nalazi u redu x i koloni y (videti organizaciju video memorije i polja atributa).

Vrednost atributa se određuje pomoću izraza:

v = z + 8 * o + 64 * s + 128 * t

gde su: z – boja zapisa – INK (0–7)
o – boja osnove – PAPER (0–7)
s – osvetljenost – BRIGHT (0–1)
t – treptanje – FLASH (0–1)

PRIMER:
```
10 PRINT AT 10,15; INK 5; PAPE
R 0; BRIGHT 1; FLASH 1;"A"
20 PRINT ATTR (10,15)
```

Pomoću linije 10 se u desetom redu i petnaestoj koloni ispisuje slovo A svetlo plave boje, na crnoj osnovi, sa povećanom osvetljenošću i sa treptanjem. Naredbom u liniji 20 se ispisuje vrednost atributa tog karaktera.

SCREEN$ (x,y)

Upotrebom ove naredbe se dobija karakter koji se nalazi u redu x i koloni y ekrana.

PRIMER:

```
10 PRINT "ekran": PAUSE 0
20 PRINT SCREEN$ (0,3)
```

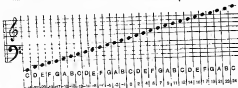
Kao rezultat se dobija karakter k koji se nalazi u nultom redu i trećoj koloni.

U slučaju da traženi karakter ne pripada Spektrumovim karakterima, rezultat će biti prazan string.

Za argumente ove naredbe važi isto što i za argumente iza kontrolnog karaktera **AT**. x je od 0 do 21, a y od 0 do 31.

BEEP t,v

Ovom naredbom se aktivira zvučnik. Prvi broj koji je naveden iza naredbe, t, određuje trajanje tona izraženo u sekundama. Drugi naveden broj, v, određuje visinu tona u polutonovima u odnosu na srednji C ton, a može biti u opsegu od − 60 do + 69.8. Trajanje i visina tona se mogu dati u obliku promenljivih ili izraza. Izvršavanje naredbe **BEEP** se ne može prekinuti.



Slika 3-3 Visina tona u zavisnosti od zadatog broja

PRIMER: **BEEP 1,2**

Upotreba tonova je naišla na najširu primenu u igrama.

PRIMER: Dobijanje zvučnog efekta koji bi mogao da se upotrebi u nekoj od igara

```
10 FOR n=-30 TO 40 STEP 7
20 BEEP .08.n
30 NEXT n
40 FOR n=50 TO -50 STEP -8
50 BEEP ABS (.005*n),n
60 NEXT n
```

Pritisak na neki od Spaktrumovih tastera može biti praćen zvukom ako se u sistemsku promenljivu PIP (23609) ubaci vrednost koja određuje trajanje tog zvuka (početna vrednost je 0).

PRIMER: **POKE 23609, 50**

Posle izvršavanja ove naredbe, pritisak na taster će biti praćen zvukom.

RAD SA KASETOFONOM

Programi i podaci se mogu u cilju kasnijeg korišćenja snimati na magnetnu traku. Po pravilu se za to koristi kaseta odnosno kasetofon.

Snimak se sastoji od dva dela: zaglavlja (engl. header) i bloka sa podacima koji mogu biti bejzik program, promenljive bajzik programa, mašinski program ili bilo kakvi podaci. Ispred zaglavlja se na traci snima ton (engl. leader) visine 1 6 KHz koji traje 5s, a ispred bloka sa podacima se snima isti ton u trajanju od 2s.

U daljem tekstu se upotrebljavaju sledeći termini:

Bajtovi – Mašinski program ili bilo kakvi podaci koji se snimaju na traku. Ovo se ne odnosi na bejzik program i višedimenzionalne promenljive iako su, naravno, i oni zapisani u obliku bajtova.

Ime – ime dodeljeno snimljenom programu ili podacima. Sa leve i desne strane imena su obavezno znaci navoda. Mora se sastojati od barem jednog karaktera, ali ne može biti duže od 10 karaktera. U protivnom se prilikom upotrebe naredbe **SAVE** javlja greška sa izveštajem **F Invalid file name**, nedozvoljeno ime bloka sa podacima.

Početak – Adresa u memoriji prvog bajta bloka bajtova

Dužina – Dužina bloka bajtova.

Slovo – Ime višedimenzionalne promenljiva. Može biti samo jedno slovo. Iza imena string promenljiva obavezan je znak $

LOAD

Ovo je naredba kojom sa programi i podaci učitavaju sa magnetne trake u memoriju računara. Obični njena primene su:

LOAD"" Ovom naredbom će se učitati, bez obzira na ime, prvi bejzik program (i njegove promenljive) koji se raprodukuje sa traka. Pri toma će bejzik progrem koji se nalazio u računaru biti izbrisan. Ako program koji se učiteva zauzime više mesta u memoriji nego što ima slobodnog prostora, odmah po učitavanju zaglavlja će se javiti greška sa izveštajem: **4 Out of memory** (nama dovoljno memorija). Pri tome se progrem koji se nalazi u računaru ne briša. Posle učitavanja zaglevlja na ekranu se ispisuje poruka: **Program:** ime.

LOAD "ime" U računar će se učitati samo bejzik program sa navedenim imenom. Ostali programi koji se reprodukuju se neće učitavati, samo ča se ispisivati njihova imena.

LOAD ""CODE Učitavanje prvih bajtova koji se reprodukuju se trake, bez obzira na ime. Smeštaju se od adresa u memoriji koja je zabeležena u zaglavlju. Broj bejtova je tekode zabeležen u zaglavlju. Stari sedržaj dela memorije u koji se smeštaju bajtovi se briše. Posle učitavanja programa na akranu sa ispisuje poruka: **Bytes:** „ima".

LOAD "ime" **CODE** Učitavaju sa samo oni bajtovi kojima je pri snimanju dodaljano navedeno ime.

LOAD ""CODE početak Bajtovi se pri učitevanju smeštaju u mamoriju od adrese koje je navedena iza naredbe. Broj bajtova koji se učitavaju je zabeležen u zaglavlju. Na primer naredbom **LOAD "" CODE 32100** učitaće se naredni blok bajtova i biče smešten u mamoriju od adresa 32100.

LOAD ""CODE početak, dužina Važi isto što i za prethodnu naredbu sem što se učitave samo navedeni broj bajtove (dužina), bez obzire na ono što ja zapisano u zeglavlju. Ako je navedeni broj bajtova različit od broje bajtova u bloku koji se učitave javlja se greška sa izveštajem **R Tapa loading error** (greške pri učitavanju sa trake).

PRIMER: **LOAD ""CODE 42359,12**

Pomoću ove naredba se učitava sa trake 12 bajtova i smešta u memoriju od adrese 42359.

LOAD ""SCREEN$ Ova naredba služi za učitavanje sadržaja TV slike. Ekvivalentna je naradbi **LOAD ""CODE 16384,6912.** To znači da se sa trake učitava 6912 bajtova i smešta u memoriju od adresa 16384

LOAD "" DATA slovo **()** Na ovaj način se učitavaju brojne ili string višedimenzionalne promenljive. Iza slova koje je ime string promenljive mora da stoji karakter $ Obavezno je stavljanje prikazanih zagrada Učitavanjem novih, brišu se stare promenljive sa istim imanom Na akranu se ispisuje poruka: **Numbar array** kada se učitava brojna višedimenzionalna promenljiva, a **Character array** kada se učitava string višadimenzionalna promanljiva.

Između znakova navoda se, u svim prethodnim naredbama, može navasti ime pod kojim su programi, promenljive ili bajtovi smeštani na traku U tom slučaju će sa učitavati samo oni sa tim imanom, bez obzira šta se reprodukuja sa trake.

Ima pod kojim se smašta na traku treba razlikovati od imena promenljivih, programa ili podataka.

PRIMER. **LOAD „matrica" DATA a$()**

Ovo je naredba za učitavanje višedimanzionalne string promenljive sa imenom a$, koja je na traku smeštena pod imenom matrica

SAVE

Pomoću ova naredbe se programi i podaci animaju na magnetnu traku (kasetu).

Iza **SAVE** se pod navodnicima obavazno navodi ime pod kojim sa programi, promenljiva i bajtovi snimaju na traku. Ako se ime na navodi ili ja duža od 10 karaktera javlja se greška sa izveštajem **F**.

Posle unošenja odgovarajućeg oblika naradba **SAVE** i pritiskanja tastera **ENTER**, računar će ispisati poruke **Start tape, then preaa any key.** Tada treba uključiti snimanje na kasetofonu i pritisnuti bilo koji tastar Kada se završi snimanja biće ispisan izveštaj **0 OK.** Tada treba isključiti kasatofon.

Oblici naredbe **SAVE:**

SAVE "ime" Na ovaj način se na traku, pod navedenim imenom, snima bejzik program.

PRIMER Prvo se unosi program

```
10 REM program primer
20 PRINT "primer"
```

a zatim se daje direktna naredbe **SAVE** "Mikro 1" Računar tada is pisuje poruku **Start the tape, then press any key** Treba uključiti snimanja na kasetofonu, pritisnuti bilo koji taster i sačekati da računar snimi program. Posle pojavljivanja izveštaja **OK**, zaustaviti kasetofon. Ovim je program iz primere snimljan na traku pod imanom Mikro 1

Ime programa se moža sastojati i od kontrolnih karaktera tako da je mogućno menjati boju i masto ispisivanja

PRIMER

```
10 SAVE CHR$ 22+CHR$ 1+CHR$ 0+
"AT 1.0"
```

Treba izvršiti navadenu liniju, e zatim učitati snimak.

SAVE "ima" **LINE** n Program koji je snimljen na ovaj način će, odmah posle učitavanja, automatski počati da se izvršava od programske linije sa brojem n. Ako broj nije naveden, izvršavanja će početi od prve linije u programu

PRIMER: **SAVE „KALENDAR" LINE 60**

Program snimljan pod imenom kalendar će, odmah kada buda učitan sa traka, početi da se izvršava od linije 60

SAVE "ima"**CODE** početak, dužina Ovom naredbom sa na traku pod navedenim imenom snima blok bajtova, odnosno sadržaj memorije, od adrese koja je jednaka prvom navedenom broju (početak) Broj bajtove koji se snima ja jednak drugom navedanom broju (dužina)

PRIMER **SAVE „slika"CODE 16384, 6912**

Pomoću data naradbe se na traci snima 6912 bajtova od adrese 16384 Konkretno, na ovaj način se čuva na traci sadržaj dela memorije koja odreduje sadržaj ekrana (video memorija i polja atributa)

SAVE "ime"**SCREEN$** Ova naradba ja specijalni slučaj gornja. Omogućeva snimanja podataka koji odreduje sadržaj ekrana. Ima identičan efekat kao nareba data u gornjem primeru.

SAVE "ima" **DATA** slovo() Snimanje brojnih i string promenljivih sa obavlja ovom naredbom Slovo navedano iza **DATA** je ime višedimenzionalne promenljive Ako je u pitanju string promanljiva iza tog slova mora da stoji karakter **$** Prikazana zagrade su obavezne

PRIMER **SAVE "brzine" DATA e()**
SAVE „tacke" DATA b$()

Ovim naredbama je na traci pod imenom brzine, sačuvana višedimenzionalna promenljiva s, a pod imenom tačke višedimenzionalne string promanljiva b$

VERIFY

Posle snimanja programa, promenljivih ili podataka na traku, poželjno je proveriti da li je program ispravno snimljan Postupak se sastoji u sledećam traka se vrati na početak snimka, upotrebi naredba **VERIFY** u odgovarajućem obliku i aktivira kasetofon Tima se sadržaj memorija upoređuje sa snimkom Ako je snimak ispravan, računar će ispisati izveštaj **OK**, a ako se pojavi izveštaj **R Tape loading error,** najbolje je ponoviti snimanja

Ako sa iza **VERIFY** postave samo navodnici, sadržaj memorije ća se uporađivati sa onim što se prvo reprodukuje sa trake, a ako sa između navodnika navede ime, upoređivanje sa vrši samo sa onim šta je snimljeno pod tim imenom

Oblici naredbe **VERIFY:**

VERIFY"" Proverava ispravnost snimka bejzik programa

VERIFY ""CODE Provera ispravnosti snimka bloka bajtova, odnosno sadržaja memorije od adrese koja je zabeležena u zaglavlju i u dužini koja je data u zaglavlju

VERIFY ""CDDE početak Provera ispravnosti snimka bloka bajtova, odnosno sadržaja mamorije, od adrase navedene iza naredbe (početak) Broj bajtova koji se proveravaju je dat u zaglavlju

VERIFY ""CODE početak, dužina.Na ovaj način se daje i početna adrasa (početak) i broj bajtova (dužina) u memoriji sa kojima se upoređuje snimak bloka bajtova

VERIFY ""SCREEN$.Naredba za upoređivanje sadržaja dela memorije u kome sa pamte podaci o TV slici i snimka tih podataka Potpuno je akvivalantna naredbi

VERIFY ""CODE 16384, 6912

Pri izvršavanju ove naredbe često se javlja greška zbog promene sadržaja ekrana.

VERIFY "" DATA slovo () Promena ispravnosti snimka višedimenzionalnih promenljivih. Ime promenljive, navedeno iza **DATA** je uvak jedno slovo iza brojne ili jadno slovo iza koga je karakter $ (za string promenljiva). Zagrade su obavezne.

U svim prathodnim slučajevima između navodnika sa može navesti ime, pa sa tada sve osim onoga što ja snimljeno pod tim imanom zanemaruje.

MERGE

Ovom naredbom je moguće učitati bajzik program bez brisanja programa koji je već u računaru. Ako postoje programske linije sa istim brojem, linija starog programa će biti obrisana i uneće se linija novog programa.

Posle učitavanja programa, izveštaj o uspešnom završetku **(OK)** se na pojavljuje odmah, već posle izvesnog vremena koja je srazmerno dužini programa.

Programi učitar' na ovaj način ne mogu da počnu da se izvršavaju automatski.

Naradba se upotrebljava samo u obliku

MERGE "ime" ili **MERGE "** jer se njome učitavaju samo bejzik programi. Ako je u navodnicima dato ime programa, moći ća da sa učita samo program snimljen pod tim imanom, a ako nema imena, učitava se prvi program koji se reprodukuje sa trake.

STRUKTURA SNIMKA NA TRACI

Već je rečeno da sa snimak sastoji iz zaglavlja i bloka sa podacima. Ispred oba dela je snimljen ton (engl. leadar) u trajanju 5s (zaglavlje) odnosno 2s (blok sa podacima).

Zaglavlje počinje bajtom čija je vrednost Ø, a blok sa podacima bajtom čija je vrednost 255 (binarno 11111111), tako da računar na osnovu toga zná kakva informacija sledi.

Zaglavlje se sastoji od sedamnaest bajtova koji sadrže sledeće podatke

Broj bajta namena

1 Njegova vrednost zavisi od vrsta snimljenih podataka i iznosi 0 – za bajzik program
1 - brojne višedimenzionalne promenljive
2 - string višedimenzionalne promenljive
3 – bajtovi (mašinski program, bilo kakvi podaci)

2–11 Sadrže ime pod kojim su program, promenljive ili podaci snimljeni Dozvoljeni su i kontrolni karakteri

12–13 Ako se na radi o bejzik programu, određuju dužinu bloka

14–15 Ako je u pitanju bejzik program određuju broj linije od koje će početi automatsko izvršavanje (ako je ta mogućnost iskorišćena) U ostalim slučajevima određuju adresu početka bloka

16–17 Sadrže dužinu bejzik programa

I zaglavlje i blok sa podacima se završavaju bajtom parnosti koji služi za otkrivanje grešaka U slučaju greške pojaviće se izveštaj **R Tape loading error.**

PODEŠAVANJE KASETOFONA

Pri snimanju programa, u slučaju da na kasetofonu postoji podešavanje nivoa snimanja, traba nivo podasiti tako da se igla na instrumentu nalazi nešto ispod oblasti označane crvenom bojom (nešto ispod 0 dB)

Kada se programi učitavaju sa traka treba kontrolu za jačinu zvuka na kasetofonu podesiti negde oko polovine Ako tada učitavanja ne uspeva (na pojavljuju se pruge na obodnom delu ekrana) potrabno je povećavati nivo Taj nivo na treba da bude ni suviše visok jer se u tom slučaju često javljaju graške

Vrlo često ja učitavanje onemogućeno zato što položaj zapisa na traci ne odgovara položaju glave za snimanje i reprodukciju u kasetofonu Problam je moguća rašiti podešavanjem visine glave okratanjem jednog od zavrtnjeva kojima je ona pričvršćena Drugi zavrtanj se obično nalazi na fiksnoj visini, pa nema uticaja na položaj glave Treba paziti da odvijač kojim sa to radi ne bude namagnetisan Zavrtanj treba okretati levo, desno, dok se ne nađe položaj u koma se najjasnija čuju visoki tonovi Ne preporučuje se podešavanje visina glave na kvalitetnim kasetofonima jer je verovatnije da je greška na traci

LLIST n

Pomoću ove naredba se na štampaču ispisuje bajzik program od linije čiji je broj n (ili od prve naradna, ako linija sa brojem n na postoji). Ako sa ne naveda nikakav broj, ispisivanje počinja od prva linija u programu.

LPRINT

Ispisivanje na štampaču se postiže naradbom **LPRINT.** Sve što ja navedano za naredbu **PRINT** važi i za naredbu **LPRINT,** sam mogućnosti ispisivanja na bilo kom mestu (ne može sa koristiti kontrolni karakter **AT**). Razlog je u tome što se podaci šalju štampaču prako memorije štampača (engl. printar buffar) koji prima sadržaj samo jadnog rada koji sa štampa.

COPY

Slika koja sa nalazi na akranu se može iscrtati na papiru pomoću odgovarajućeg štampača upotrabom naredba **COPY**. Zapisu **(INK)** na ekranu odgovara zapis (tačka) na papiru.

Sledeće naradbe se odnose na rad sa mikrodrajvom i način biti objašnjenja u ovoj knjizi.

CAT
CLOSE
DELETE f
ERASE
FORMAT f
MOVE f_1, f_2
OPEN

3–3 IZVEŠTAJI

Izveštaji se pojavljuju u donjam delu ekrana kada računar prestane da izvršava bejzik program, objašnjavajući razlog, bez obzira da li je prekid nastao uslad graške ili ne.

Svaki izveštaj počinje brojam ili slovom, što predstavlja njegov kôd. Zatim sladi kratka poruka. Posla nje se ispisuje broj linije i broj mesta u toj liniji na kome se nalazi naradba u kojoj je došlo do prestanka izvršavenja.

U tabeli su navedni izveštaji i okolnosti pod kojima sa javljaju.

KÔD	ZNAČENJE	SITUACIJA
0	OK Uspašan završetak ili prelazak na programsku liniju čiji je broj veći od bilo koje postojeće	bilo koja
1	NEXT without FOR Na postoji kontrolna promenljiva (ona koja se navodi iza FOR) ili postoji obična promanljiva sa tim imenom, a na kontrolna koja se definiše iza FOR.	NEXT
2	Variable not found Nije pronađena promanljiva. Kada je u pitanju obična promenljiva to će se dasiti kada joj nije dodeljena vrednost pomoću naredbi LET, READ, INPUT, iza FOR ili nije učitana sa traka. Za višedimenzionalne promanljiva ova greška se javlja kada nije demenzionisana naradbom DIM ili nija učitana sa trake.	bilo koja
3	Subscript wrong Pogrešan indeks Indeks je veći od dimenzija promenljive ili je u indeksu pogrešan broj. Ako je indeks negativan ili veći od 65535 dobija se greška sa izveštajam B.	višedimenzionalna promenljiva i podstringovi
4	Out of mamory Može se desiti da zbog ovog razloga računar počna da se ponaša neregularno. Tada trabа izbrisati sve iz editorska linije pomoću DELETE i zatim izbrisati još nešto iz programa	LET, INPUT, FOR, DIM, GO SUB, LOAD, MERGE, za vreme izračunavanja izraza

	keko bi se oslobodilo nešto memorije za normalan rad.	
5	Out of screen Javlja se pri pokušaju da se nešto napiše van mesta na ekranu koja su za to određene. Pomoću naredbe INPUT se pokušava dodeljivanje više od 23 reda donjem delu ekrena ili se pokušava ispisivanje u donjem delu ekrana pomoću naredbe PRINT AT (npr. PRINT AT 22,...)	INPUT, PRINT AT
6	Number too big Suviše veliki broj. Izračunavanjime se došlo do broja većeg od 1.7×10^{38}	Aritmetičke operecije
7	RETURN without GOSUB Nailazek ne naredbu RETURN za koju prethodno nije deta naredba GOSUB.	RETURN
8	End of file	mikrodrejv
9	STOP statement Izvršena je naredba STOP. Naredbom CONTINUE može se nastaviti izvršavenje programa od sledeće naredbe (iza STOP).	STOP
A	Invalid argument Nije dober argument funkcije	SQR, LN, ASN, ACS, USR slovo.
B	Integer out of renge Intidžer je van opsega Do ove greške dolazi i keda je argument koji je broj se pomičnim zarezom zaokružen na intidžer koji je van opsega.	RUN, RANDOMIZE, DIM, POKE, GO TO, GO SUB, LIST, LLIST, PAUSE, PLOT, CHR$, PEEK, USR broj

C	Nonsense in BASIC Tekst argumenta ili string argumenta nije pravilan izraz.	VAL, VAL$
D	BREAK-CONT repeats Naredba BREAK je data u toku rada sa periferním jedinicama. Naredba CONTINUE će dovesti do ponovnog izvršenja naredbe u kojoj ja došlo do prekida.	LOAD, SAVE, VERIFY MERGE, LPRINT, LLIST COPY i kada se posle pitanja scroll? pritisne N, SPACE ili STOP.
E	Out of DATA Pokušaj da sa učita više podataka nego što ih ima iza naredba DATA.	READ
F	Invalid file name Iza naredbe SAVE nema imena ili je duže od 10 karaktera	SAVE
G	No room for line U memoriji nema dovoljno mesta za novu programsku liniju.	ubacivanje linija u programu
H	STOP in INPUT Za vreme izvršavanja naredbe INPUT je pritisnuto STOP ili za vreme izvršavanja naredbe INPUT LINE (CAPS SHIFT i 6). Naredbom CONTINUE se nastavlja izvršavanje ponavljajući naredbu INPUT (za razliku od slučaja sa izveštajem 9).	INPUT
I	FOR without NEXT Ne postoji naredba NEXT za navadanu naredbu FOR	FOR
J	Invalid I/O device	Mikrodrajv
K	Invalid colour Broj navedan iza naredbe nema odgovarajuću vrednost	INK, PAPER, BORDER, FLASH, BRIGHT, INVERSE, OVER, ili iza odgovarajućih kontr. karaktera

L	BREAK into program Izvršena je naredbe BREAK. Do prekida dolazi između dve naredbe. Broj naredbe neveden u izveštaju se odnosi na naredbu izvršenu pre prekida. Pomoću CONTINUE se prelazi na izvršavenje sledeće naredbe.	bilo koja
M	RAMTOP no good Broj koji se dodeljuje sistemskoj promenljivoj RAMTOP je suviše velik ili suviše mali.	CLEAR, RUN
N	Statement last Prelazak na izvršavanje naredbe koja više ne postoji.	RETURN, NEXT CONTINUE
O	Invalid stream	mikrodrajv
P	FN without DEF Funkcija FN nije prethodno def. pomoću DEF	FN
Q	Parameter error Pograšan broj argumenate, ili je neki od njih pograšnog tipa, string umesto broja i obrnuto.	FN
R	Tape loading error Greška pri učitavanju se trake. Podaci na traci su nađeni, ali iz nekog razloga ne mogu biti učitani ili su pogrešno učitani.	VERIFY, LOAD, MERGE

3–4 SISTEMSKE PROMENLJIVE

Sistemske promenljive su sadržaji memorijskih lokacija RAM–a sa edresama od 23552 do 23733. One svojom vrednošću ukazuju na izvesna stanja u toku reda računara. Monitorski program u svom radu koristi postavljene, e i postavlja nove vrednosti sistemskih promenljivih. Treba razlikoveti sistemske od programskih promenljivih.

Pogodnom upotrebom i manjanjem sistemskih promenljivih korisnik može znatno proširiti mogućnosti računara i pojednostaviti programiranja.

Očitavanje i postavljanje novih vrednosti jednobajtnih promenljivih se ostvaruje **PEEK** ili **POKE** naredbom. Važna promenljivih je dvobajtna. Tada treba uočiti da bajt na višoj adresi treba da ima 256 puta veću težinu. Očitavanje vrednosti takve promanljive se obavlja na sledeći način

PRINT PEEK n+256*PEEK (n+1)

gde je n adrasa promenljive tj. adresa bajta manje težina

Ubacivanje nove vrednosti dvobajtne promenljive se obavlja naredbom

POKE n, v–256*INT (v/256): POKE n+1, INT (v/256) gde je **v** nova vrednost koju uzima promenljiva.

Sledeća tabala daje pragled svih sistemskih promanljivih, adresa od koja počinju, broj bajtova koje zauzimaju, ime dato od Sinclair Research i komentar

Adresa dacimal-no	heks	Broj bajtova	Ime	Komentar
23552	5C00	8	KSTATE	Dve grupa, po četiri bajta koji se koriste za očitavanje stanja na tastaturi i ispitivanje funkcije ponavljanja kada je taster duže pritisnut. Ova dve grupe dozvoljavaju detekciju pritiska novog tastera u periodu ponavljanja prethodnog tastera. Rezultat svega se stavlja u LAST K.
23560	5C08	1	LAST K	U nju se smešte kôd poslednjeg pritisnutog tastera. (Obratiti pažnju na peti bit promenljive 23611)
23561	5C09	1	REPDEL	Vreme (izraženo u 1/50 sekunde) u toku koga taster traba da bude pritisnut da bi počela funkcija ponavljanja Spektrum

				postavlja tu vrednost ne 35 (0,7s) eli se može upisati bilo koja.
23562	5C0A	1	REPPER	Vreme (u 1/50s) između ponavljanja funkcije tastera. Zedata početne vrednost iznosi 5 (1/10s)
23563	5C0B	2	DEFADD	U ovu promenljivu se stavlja adresa argumenta bejzik naredbe **DEF FN**. Primenjuje se pri prenošenju vrednosti iz bejzik progreme u mašinski program.
23565	5C0D	2	K DATA	Koristi se za čuvanje drugog bejta boje unešene direktno sa tastature.
23566	5C0E	2	TVDATA	Čuva bajtove boje, **AT** i **TAB** kontrole.
23568	5C10	X38	STRMS	Adrese kanala povezanih sa strimovima.
23606	5C36	2	CHARS	Za 256 manja od adrese tabele karektera. Može se upotrebiti za uvođenje sopstvenog seta karaktere.
23608	5C38	1	RASP	Trajanje upozoravajućeg zvuka, koji se javlja pri nedostatku prostora u memoriji ili pri više od 255 naredbi u programskoj liniji
23609	5C39	1	PIP	Trajanje tonskog odzive testeture, početna vrednost iznosi nula
23610	5C3A	1	ERR NR	Vrednost za jedan manje od koda izveštaja. Koristi se od strene ROM rutine RST 08h
23611	5C3B	X1	FLAGS	Bitovi promenljive se koriste za razne kontrole bejzika. Peti bit je od interesa jer ukazuje da li je pritisnut novi taster

23612	5C3C	X1	TV FLAG	Upotrebljava se u kontroli formiranja TV slike.
23613	5C3D	X2	ERR SP	Koristi se uz ROM rutinu RST Ø8h. Ukazuje na memorijsku lokaciju u kojoj se nalazi adresa nastavljanje mašinskog programa u slučeju greške.
23615	5C4F	2	LIST SP	Povratna adresa nakon izvršenja naredbe LIST
23617	5C41	1	MODE	Definiše pokazivač: K, L, C, E ili G.
23618	5C42	2	NEWPPC	Broj programske linije na koju se prelazi.
2362Ø	5C44	1	NSPPC	Broj naredbe u liniji na čije izvršenje se prelazi. Upisivanjem konkretnih vrednosti u NEWPPC i NSPPC program forsirano kreće od zedatog mesta.
23621	5C45	2	PPC	Broj progremske linije u kojoj se nalazi naredba koja se izvršava.
23623	5C47	1	SUBPPC	Broj mesta u liniji naredbe koja se izvrševa.
23624	5C48	1	BORDCR	Boja obodnog dela ekrena, puta osam, ista kao i za donji deo ekrana (editorske linije).
23625	5C49	2	E PPC	Broj programske linije označene pokazivačem.
23627	5C4B	X2	VARS	Adresa odakle počinju programske promenljive.
23629	5C4D	X2	DEST	Adresa promenljive pri dodeli vrednosti.
23631	5C4F	X2	CHANS	Adresa kenalnih podataka.
23633	5C51	X2	CURCHL	Adresa podataka koji se trenutno koriste za komunikaciju sa periferним jedinicama.
23635	5C53	X2	PROG	Adresa bejzik programe (menja se upotrebom mikrodrajva).

23637	5C55	X2	NXTLIN	Adresa sledeće linije u programu
23639	5C57	X2	DATADD	Adresa kraja podataka u DATA naredbi.
23641	5C59	X2	E LINE	Adresa naredbi koje se unose preko tastature.
23643	5C5B	2	K CUR	Adresa pokazivače.
23645	5C5D	X2	CH ADD	Adresa sledećeg bejzik karaktera koji treba da se interpretira.
23647	5C5F	X2	X PTR	Adresa karaktera posle znaka ?
23649	5C61	X2	WORKSP	Adrese privremenog radnog prostora
23651	5C63	X2	STKBOT	Adresa početka računarskog prostora (računarskog steka)
23653	5C65	X2	STKEND	Adresa kreja računarskog prostora i početka slobodnog prostora.
23655	5C67	1	BREG	Upotrebljeno od strane B registra za rad ROM programa ze računenje
23656	5C68	2	MEM	Adresa prostora koji se koristi za računanje.
23658	5C6A	1	FLAGS2	Bajt opšte namene. **POKE** 23658,8 prevodi računar u C način rade
23659	5C6B	X1	DF SZ	Broj linija donjeg dele ekrana (editorskog prostore) Zadata vrednost je 2. Ubacivanjem vrednosti 0 može se proširiti ispisivanje preko celog ekrana (obavezno vratiti prethodnu vrednost).
23660	5C6C	2	S TOP	Broj progremske linije od koje se lista program.
23662	5C6E	2	OLDPPC	Broj linije na koju se odlazi naredbom **CONTINUE**.
23665	5C71	1	FLAGX	Bejt opšte namene

23666	5C72	2	STRLEN	Dužina unesenog stringa.
23668	5C74	2	T ADDR	Adrese narednog člane u tabeli sintakse.
23670	5C76	2	SEED	Početne vrednost iz koje se izračunava slučajni broj. Postevlja se naredbom **RANDOMIZE.RANDOMIZE** N če dati vrednost N ovoj promenljivoj.
23672	5C78	3	FRAMES	Broj poluslika (TV), prikazanih od trenutka uključenja računara. Povećava se svakih 20 ms 3 bajta omogučavaju brojanje do oko 9,2h nakon čega se brojanje ponavlje.
23675	5C7B	2	UDG	Adresa prvog bajte prostora za grafiku definisenu korisnikom.
23677	5C7D	1	COORDS	x-koordineta poslednje PLOT tačke.
23678	5C7E	1		y-koordinata poslednje PLOT tečke.
23679	5C7F	1	PR CC	Bajt manje težine adrese sledeće LPRINT pozicije na štampaču.
23680	5C80	1	P POSN	Broj kolone u radu se printerom.
23681	5C81	1		Neupotrebljeni bejt
23682	5C82	2	ECHO E	Krajnji broj kolona i redova donjeg dela ekrana.
23684	5C84	2	DF CC	Adresa **PRINT** pozicije na ekranu.
23686	5C86	2	DFCCL	Adresa **PRINT** pozicije u donjem delu ekrana.
23688	5C88	X1	S POSN	Broj kolona za naredbu **PRINT**.
23689	5C89	X1		Broj redova za naredbu **PRINT**.
23690	5C8A	X2	SPOSNL	Isto kao S POSN, eli u donjem delu ekrana.
23692	5C8C	1	SCR CT	Broj pomeranja (scroll) slike Uvek je za 1 veči od broja po-

				meraja slika pre pitanja **scroll?** Upisivanjem broja većeg od 1, doći će do pomeranja slike bez pitanja.
23693	5C8D	1	ATTR P	Stalna tekuća boja ekrana.
23694	5C8E	1	MASK P	Upotrebljava se za funkciju transparencije boje (INK 8). Bitovi postavljeni na 1 maskiraju bitove promenljive ATTR P i omogućuju upotrebu odgovarajućih atributa, onoga što je već prikazano na ekranu.
23695	5C8F	1	ATTR T	Privremeno postavljena tekuća boja.
23696	5C90	1	MASK T	Isto kao MASK P, ali za ATTR T.
23697	5C91	1	P FLAG	Bajt opšte namene.
23698	5C92	30	MEMBOT	Memorijski prostor dela za računanje. Smeštaju se brojavi koji ne mogu uobičajeno biti stavljeni na računarski stek.
23728	5CB0	2		Dva neupotrebljena bajta.
23730	5CB2	2	RAMTOP	Adresa poslednjag bajta za rad sa bejzikom.
23732	5CB4	2	P-RAMP	Adresa poslednjeg bajta memorija.

U promenljive sa oznakom X ne treba ništa upisivati, jer bi lako moglo doći do poremećaja rada računara.

3-5 MAPA MEMORIJE

U svim verzijama Spektruma se nalazi ROM memorija kapacitèta 16K u kojoj su zapisani operativni sistem i bejzik interpreter.

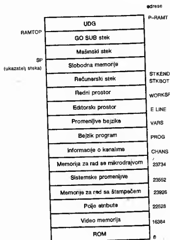
Postoje dve varijanta Spektruma: sa 16K RAM memorije i sa 48 K RAM memorija. Sada će se prikazati u kojim delovima memorije su smeštene pojedine vrste podataka. Sa leve strana date su adrese od kojih počinje ona vrsta podataka koja je navedena sa desne strane. Na pojedinim mestima je umasto adrese dato ime sistemske promenljive u kojoj je smeštena potrebna adresa.

adresa	podaci
Ø	– ROM
16384	– Video memorija, podaci o sadržaju slike (engl. display file).
22528	– Atributi, informacija o boji, osvetljenju i treptanju (detaljnije o tome u sledećem poglavlju)
23926	– Memorija za rad sa štampačem (engl.printer buffer). Sadrži podatke o 32 karaktera koji se šalju na štampač. Ako štampač nije upotrebljen, može se koristiti u druge svrhe.
23552	– Sistemske promenljive
23734	– Podaci koji se koriste pri radu sa mikrodrajvom.
CHANS	– Informacije o kanalima koji služe za razmenu podataka se periferním jedinicama.
PROG	– Bejzik program. Ova adresa je obično 23755
VARS	– Promenljive bejzik programa
E LINE	– Sadržaj editorskih linija. To su direktna naredba i linije koje se upisuju ili ispravljaju. Veličina ovog prostora se menja sa promenom sadržaja aditorskih linija.
WORKSP	– Podaci koji se unose naredbom **INPUT** i podaci koji se koriste pri sabiranju stringova
STKBOT	– Računarski stek, privremana memorija koja se koristi pri izračunavanjima.
STKEND + 1	– Slobodni prostor
Ukazivač staka u Z80	– Mašinski stek, deo memorije koji mikroprocesor Z80 koristi za privremeno smeštanje podataka
	– GOSUB stek, služi za čuvanje podataka o mestu povratka posle izvršavanja potprograma. Ta informacija ja smeštena u 3 bajta. U prvom je broj naradba unutar linije, a u sledeća dva, broj linije na koju se vraća.
RAMTOP	– Počinju podaci koji nisu dostupni bejzik programu

UDG – Podaci o karakterima (grafici) koje definiše korisnik.

P-RAMT – Poslednji bajt u memoriji.

Preglednija mapa memorije je data na slici 3-4.



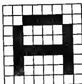
Slika 3–4 Mapa memorije

3-6 ORGANIZACIJA VIDEO MEMORIJE I POLJA ATRIBUTA

Za formiranje TV slike Spektrum koristi 6144 bajta na adresama od 16384 do 22527 koji čine video memoriju. 768 bajta na adresama od 22528 do 23925 služi za pamćenje atributa, podataka koji određuju boju, sjajnost i traptanje karaktera. Ta oblast mamorije se naziva polja atributa.

ORGANIZACIJA VIDEO MEMORIJE

Slika sa sastoji od 24 reda sa po 32 karaktera. Svaki karakter ja formiran od 8 x 8 tačaka (piksela). Za pamćenja jednog karaktera potrebno je 8 bajta odnosno 64 bita, jer svakoj tački na ekranu odgovara jedan bit u mamoriji. Ako je bit postavljen na jedinicu, na odgovarajućem mestu na ekranu postoji tačka, a u suprotnom ne postoji. Ukupan broj tačaka slika (rezolucija) koju Spektrum generiše na TV ekranu ja 256 x 192.



Slika 3-5 Karakter slova A

Sada ce biti objašnjeno kojim redom se stanja pojedinih tačaka na ekranu pamti u video memoriji. 24 reda karaktera je podeljeno u 3 grupe od po 8 redova. Prvo se pamti gornja pa srednja, a zatim donja grupa. Prvi bajt video memorije (adresa 16384) odgovara gornjem prvom redu tačaka prvog karaktera. Sledeći bajt odgovara prvom redu tačaka drugog karaktera u prvom redu i tako sve do poslednjeg (32) karaktera u prvom redu (adrasa 16415). Sledeća 32 bajta odgovara prvom gornjem redu tačaka drugog rada karaktera

Zatim sleda oni koji odgovaraju gornjem redu tačaka tekućeg rada karaktera i tako sve do osmog rada karaktera. Posle njih dolaze bajti koji odgovaraju drugom redu tačaka prvog reda karaktera, zatim oni koji odgovaraju drugom redu tačaka drugog reda karaktera, pa drugom redu tačaka trećeg reda karaktera i tako radom.

Na isti način se pamta podaci o srednjoj grupi redova, a za njom i podaci o donjoj grupi.

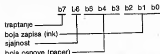
PRIMER: Izložena organizacija video memorije sa može uočiti tako što se vidao mamorija redom popunjava jadinicama. To se može postići uz pomoć sledaće programske linija

10 FOR n=16384 TO 22527:POKE n,255: NEXT n

Zamenom vrednosti 255 (1111 1111) nakom drugom vradnošću ekran naće biti sasvim popunjen.

ORGANIZACIJA ATRIBUTA

Svakom karakteru odgovara jedan bajt, atribut, pomoću koga sa definiše boja osnove, boja zapisa, osvetljenost i treptanje. Ti podaci su zabeleženi na sledeći način:



biti od b0 do b2 određuju boju zapisa (engl.ink), bitovi od b3 do b5 boju osnove (angl.paper), bit b6 sjajnost karaktera, a bit b7 da li karakter trapti ili ne.

PRIMER Upisivanjem u atribut broja 113 (u binarnom obliku to je 01110001) odgovarajući karakter će imati piavu boju zapisa 001 , žutu boju osnove (110), povaćanu sjajnost (1) i neće treptati (0).

Na adresi 22528 se nalazi atribut prvog karaktera u prvom redu na ekranu. Sledeđeći bajt ja atribut drugog karaktara u prvom redu, zatim sledi atribut trećeg karaktera i tako do adrese 22559 gde se nalazi atribut poslednjeg karaktera prvog reda. Posle toga slede

atributi karaktera drugog reda, pa trećeg reda i tako redom do adrese 23925 gde je smeštan atribut poslednjeg karaktera u poslednjem redu

PRIMER Promenićemo atribut trećeg karaktera u petom redu na ekranu pomoću naredbe

1∅ POKE 22528+32*(5-1)+(3-1), BIN 11110001

Tada on trepti, ima povećanu sjajnost, ima žutu boju osnove i plavu boju zapisa

3-7 SMEŠTANJE BEJZIK PROGRAMA

Početak dela memorije u kome se nalazi bejzik program određen je sadržajem sistemske promenljive PROG (adrese 23635 i 23636). U standardnoj Spektrumovoj konfiguraciji, bejzik program počinje od memorijske lokacije 23755. Početna adresa se menja prilikom upotrebe mikrodrajva ili proširenjem područja podataka kanala.

Korisno je pogledati kako izgleda jedna bejzik linija smeštena u memoriji. To se može izvršiti upotrabom programa datog na kraju dela 2-5.

Prva dva bajta svake programska linije sadrže broj programske linije. Vrednost prvog bajta ima težinu 256, a drugog 1. Na primer, za liniju broj 3∅∅ vrednost sadržaja prvog bajta je 1, a drugog 44. Treći i četvrti bajt sadrže dužinu programske linije izraženu u bajtima pri čemu je uzeta u obzir i naredba **ENTER**. Treći bajt je težine 1, a četvrti 256. Nakon toga dolaze bajti u kojima je sadržana bajzik linija. U jednom bajtu je sadržan kôd karaktera ili ime naredbe.

Izuzetak su brojevi jer se njihova vradnost nalazi u dodatnih pet bajta u obliku petobajtne forme. Bajt sa sadržajem 14 ukazuje da sledi vrednost broja.

3-8 PRINCIPI PROGRAMIRANJA

U ovom poglavlju su iznete osnovne postavke i principi programiranja.

Svaki program se može pradstaviti pomoću tri osnovne programske strukture. To su: – sekvenca
– selekcija
– itaracija

Sekvenca, ili linijska struktura, predstavlja niz naredbi koje se izvršavaju jedna za drugom.

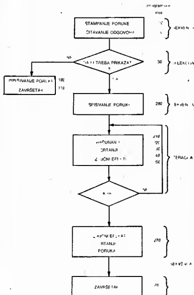
Selekcija, ili razgranata struktura, predstavlja mesto grananja u programu. Na Spektrumovom bejziku to se ostvaruje naredbom **IF...THEN**...

Itaracija, ili petlja, je struktura u kojoj se jedna ili više naredbi više puta za redom izvršavaju, sve dok ne bude zadovoljan uslov izlaska iz petlje. Na Spektrumu sa ostvaruje naradbama **FOR...NEXT**...ili **IF...THEN**... uz pomoć naradbe **GO TO.**

Način na koji se neki problem rešava, algoritam, često je pogodno predstaviti u obliku dijagrama toka. On se sastoji od međusobno povezanih osnovnih struktura.

PRIMER: Dati su program i dijagram toka kojim se taj program može predstaviti.

```
 10 PRINT AT 10,1;"Da li zelite
da vam"," nesto pokazem ?"
 20 INPUT a$
 30 IF a$="da" THEN  GO TO 200
 100 PRINT AT 15,8;"Dobro,onda z
dravo"
 110 PAUSE 120: CLB : STOP
 200 CLS : PRINT AT 12,12;"Odlic
no !": PAUSE 80: CLS
 210 LET n=20
 220 LET a=(n-87)*(n>87)
 230 PLOT 0,87: DRAW 127-a,n-a:
DRAW 127,-n
 240 BEEP .002,n/3
 250 IF n<157 THEN  LET n=n+3: G
O TO 220
 260 BEEP .1,60: DRAW -182,71
 270 PRINT #0;AT 0,12;"HVALA": P
AUSE 0
 280 STOP
```


ŠTAMPANJE PORUKE
ČITAVANJE ODGOVORA
NE
TREBA PRIKAZAT
ZAVRŠETAK
100
110
30
SPISIVANJE PORUKE
200
CRTANJE
210
40
NE
RTANJE
PORUKA
270
ZAVRŠETAK

FAZE PRAVLJENJA PROGRAMA

Pravljenje programa se ne sastoji samo od pisanja naredbi, već se ceo postupak odvija u više faza:

- sagledava sa problem i definišu ulazne i izlazne veličine nalazi se način rešavanja problema, algoritam (moguće je formiranje dijagrama toka)
- piše se program
- program se unosi u računar
- program se pušta u rad pri čemu se otklanjaju greške i vrši sa njegovo testiranje
- pravi se snimak programa
- piše se završna dokumentacija

Dokumentacija koja sa pravi u svim fazama i završna dokumentacija traba da učina program jasnim i razumljivim samom programeru i drugim programarima, što je posebno značajno, ako je potrebno program izmeniti ili pronaći greške.

MODULARNOST PROGRAMA

Velike i složena programa treba razložiti u više modula, odnosno nekoliko manjih celina. Treba da postoji glavni program koji je relativno jednostavan i čiji je glavni zadatak da poziva module kada je to potrebno. Same module je opat moguće podeliti u viša jednostavnijih calina, module nižeg stepana. Ovakva organizacija olakšava pisanje programa, jer sa stvara modul po modul, a moguće ja i raspodeliti zadatke drugim programerima. Svaki modul posebno može da se testira i pusti u rad i na kraju da se svi zajedno uklope. To je moguće čak iako naki od modula nisu završeni, a postiža sa njihovim simuliranjem pomoću nekoliko naredbi.

BRZINA IZRŠAVANJA PROGRAMA I UTROŠAK MEMORIJE

Ovo su dve kritične veličine koje kazuju da li neki program uopšte može da se izvrši i koliko je to izvršavanje efikasno.

Brzina programa se postiže njegovim skraćivanjem i biranjam najpogodnijih naredbi. Naročito su kritične naredbe koje se izvršavaju u nekoj petlji. Izračunavanja koja namoraju da se nalaze u samoj petlji treba staviti van nje. Vrema izvršavanja programa dosta zavisi od komuniciranja sa periferním uređajima, pa takve postupke treba optimizirati. U slučaju Spektruma to se posebno odnosi na rad sa mikrodrajvom ili diskovima.

Memorija se štedi pisanjem što kraćeg programa i dobrom organizacijom smeštanja podataka, što je posebno kritično pri radu sa vektorima i matricama.

Programi pisani na mašinskom jeziku su veoma efikasni, ali ih je teže pisati nego programe na bejziku. Često je pogodno napraviti kompromis tako da se glavni program i pojedini delovi pišu na bejziku, a pojedini potprogrami na mašinskom jeziku.

3-9 PRIMERI BEJZIK PROGRAMA

PRIMER 1

Program SLOVOPAD je jedna zanimljiva igra u kojoj igrač hvata u koš slova koja padaju. Koš se nalazi u donjem delu ekrana i može se pomerati levo i desno pomoću tastera 1 i 0. Uhvaćena slova se stavljaju u za njih predviđena mesta.

Linijom 10 se definišu slova koja će padati. To su slova reči Spektrum i mogu biti zamenjena nekim drugim. U ovoj liniji se takođe definišu dva UDG karaktera. To se ostvaruje petljom u kojoj se očitavaju vrednosti navedene iza naredbe DATA u liniji 600. Tim vrednostima se pune 16 bajtova UDG polja koji odgovaraju slovima a i b. Na taj način će se pod slovom b dobiti koš, a pod slovom a mesta u koja se stavljaju uhvaćena slova. Na kraju ove linije se definiše i početna koordinata uk.

Linijom 50 se postavlja brojač slova b, a **FOR** petljom se iscrtava potreban broj mesta za smeštanje slova. Slovo a (mesto smeštanja) dato pod navodnicima u ovoj liniji treba uneti nakon što se pokazivač prebaci u oblik G.

U sledećoj liniji se definišu brojač slova b i slovo koje će padati p$. U liniji 110 se slučajnim izborom određuje y koordinata slova u padu. Dobijena koordineta može imati vrednosti između 8 i 23.

Od linije 120 do linije 160 je petlja u kojoj se iscrtava slovo u padu. Pri tome se odlazi na potprogram na liniji 500 u koma se testira pritisnutost tastera 1 i 0 i koriguje koordinate koša. U liniji 150 slovo b (koš) treba upisati sa pokazivačem prebačenim u oblik G. To isto važi i za liniju 310.

Testiranje da li je slovo uhvaćeno obavlja se u liniji 170. U slučaju da ja slovo uhvaćeno, u petlji od linije 300 do linije 330, čeka sa da sa slovo postavi na odgovarajuća mesto. Nakon toga se u liniji 340 proverava da li je u pitanju poslednje slovo. Ako nije, prelazi se

* Spektrum pr[illegible]

na novo slovo odlaskom na liniju 100, a ako jeste, izvršavaju se linije 400, 410 i 420 predviđene za kraj.

Težina igre se može menjati promenom granica y koordinate slova u padu u liniji 110.

```
 10 LET w$="Spektrum": FOR n=0
TO 15: READ d: POKE USR "a"+n,d:
 NEXT n: LET yk=16
 50 CLS : LET b=0: FOR n=1 TO L
EN w$: PRINT #0;AT 1,1+n;"a": NE
XT n
 100 LET b=b+1
 110 LET p$=w$(b): IF p$="" THEN
 GO TO 100
 120 LET y=8+INT (16*RND)
 130 FOR n=0 TO 21
 140 PRINT AT n,y; INK 6*RND;p$:
BEEP .01,10-n
 150 GO SUB 500
 160 PRINT #0;AT 0,yk;" b "
 170 NEXT n
 180 IF yk=y-1 THEN  GO TO 300
 200 LET p$=" Probajte ponovo  "
 210 FOR n=1 TO 17: PRINT AT n,y
k;p$(n): BEEP .2,10-n: NEXT n: G
O TO 50
 300 GO SUB 500
 310 PRINT #0;AT 0,yk;" b "
 320 IF yk=b THEN  PRINT #0:AT 1
,b+1;p$: BEEP .1,-10: GO TO 340
 330 GO TO 300
 340 IF b<>LEN w$ THEN  GO TO 10
0
 400 FOR 1=1 TO 5: FOR n=7 TO 14
: PRINT AT n,13; PAPER 8*RND; IN
K 9;"BRAVO": BEEP .01,n: NEXT n:
 NEXT 1
 410: BORDER 6: BORDER 5: BORDER
 4: BORDER 3: BORDER 2: BORDER 1
```

```
: BORDER 0: BORDER 7: IF INKEY$=
"" THEN  GO TO 410
  420 STOP
  500 IF INKEY$="1" THEN  LET yk=
yk-1: IF yk<0 THEN  LET yk=0
  510 IF INKEY$="0" THEN  LET yk=
yk+1: IF yk>29 THEN  LET yk=29
  520 RETURN
  600 DATA 0,0,0,0,0,129,129,255,
213.171,86,106,52,44,52,44
```

PRIMER 2

Program BITOSABIRAČ prikazuje sabiranje binarnih brojeva. Izvršavanjem programa od linije 200 (RUN 200) pred korisnika se stavljaju dve mogućnosti: 1-vežba, u kojoj se sabiraju uneti binarni bojevi i 2-provera, pri kojoj korisnik vrši sabiranje uz kontrolu od strane računara.

Od linije 10 do 50 se nalazi potprogram koji obezbeđuje upisivanje, brisanje i ispisivanje binarnih brojeva, a takođe i ispisivanje decimalne vrednosti binarnog broja. Dve petlje od linije 220 do 240 i od linije 310 do 330 obezbeđuju unošenje prvog odnosno drugog broja. U petlji od linije 430 do 460 se ispisuje zbir u binarnom obliku koji se nalazi iz decimalne vrednosti u liniji 430.

Linija 450 se izvršava samo u slučaju provere i pri tome se proverava ispravnost pritisnutog tastera. Konačno u liniji 470 se utvrđuje vrednost poslednjeg bita, a linijom 490 se ponovo startuje program.

Posle razumevanja rada programa, čitaocu neće biti teško da izmenama u linijama 20, 40 i 400 omogući i sabiranje brojeva sa više od 8 bita.

```
   10 LET t$=INKEY$
   20 IF (t$="1" OR t$="0") AND L
EN v$<9 THEN  BEEP .05,2: LET v$
=v$+t$: LET v=2*v+VAL t$
   30 IF t$="9" AND LEN v$>1 THEN
 BEEP .05,-10: LET v$=v$( TO LE
N v$-1): LET v=INT (v/2)
   40 PRINT AT x1,20-LEN v$;v$;TA
B 24-LEN STR$ v;v
```

```
 50 RETURN
 200 LET p=0: PRINT AT 15,0:"1-v
ezba"'"2-provera": PAUSE 0: IF I
NKEY$="2" THEN  LET p=1
 205 CLS : PRINT AT 17,0;" koman
de"'"1"'"0"'"9-brisanje"'"ENTER"
 210 LET v$=" ": LET v=0: LET x1
=10
 220 GO SUB 10
 230 IF CODE t$=13 THEN  BEEP .1
,0: LET a$=v$: LET a=v: GO TO 30
0
 240 GO TO 220
 300 LET v$=" ": LET v=0: LET x1
=11
 310 GO SUB 10
 320 IF CODE t$=13 THEN  BEEP .1
,0: GO TO 400
 330 GO TO 310
 400 PLOT 93,75: DRAW 107,0: PRI
NT AT 11,10:"+"
 410 LET v=v+a: PRINT 'TAB 24-LE
N STR$ v;v
 420 LET a=19
 430 LET v1=INT (v/2): LET v=v-2
*v1: PAUSE 25: IF p=0 THEN  GO T
O 460
 450 PAUSE 0: IF INKEY$<>STR$ v
THEN  PRINT AT 18,18;"pogresno":
 BEEP .2,-15: PRINT AT 18,18:"
        ": GO TO 450
 460 PRINT AT 13,a:v: BEEP .06,2
: LET v=v1
 470 IF v<>0 THEN  LET a=a-1: GO
TO 430
 480 PRINT AT 18,20;"p-ponovo"
 490 PAUSE 0: IF INKEY$="p" THEN
 RUN 200
 500 GO TO 490
```

Programiranje na mašinskom jeziku

4

Mala brzina izvršavanja bejzik programa i veličina memorije koju bejzik program zauzima glavni su razlozi za programiranje na mašinskom jeziku. Ta ograničenja naročito dolaze do izražaja u velikim programima, u programima za upravljanje, regulaciju, merenje, crtanje složenih crteža i obradu velikog broja podataka i razultata. Prednost bajzik programa je lakoća i preglednost u njegovom pisanju.

Pisanjam programa na osnovnom jeziku računara, mašinskom jeziku, obezbeđuje se povećanje brzine i smanjenje potrošnje memorije u odnosu na odgovarajući bejzik program. Sa druge strane, pisanje, proveravanje i nalaženje grašaka u mašinskom jeziku je teža, što zahteva dodatna znanja i iskustva.

Formiranje programa kombinovanjem bejzik i mašinskog jazika predstavlja najbolje rešanja koja objedinjuje pradnost oba jezika.

Spaktrum ROM sadrži veliki broj programa (rutine) pisanih na mašinskom jaziku. Oni se mogu koristiti u raznim programima čime sa eliminiša potreba za njihovim razvijanjem.

U ovom delu knjige čitalac će biti postupno upoznat sa onim što je potrebno za uspešnc stvaranje mašinskih programa.

4-1 OD BEJZIKA DO MAŠINSKOG PROGRAMIRANJA

Bejzik program ja sačinjem od bejzik programskih linija. Svaka linija ima svoj broj koji joj određuje položaj u programu i bejzik naredbu za kojom moža da sledi neki podatak. Bejzik programska linija

10 LET A = 1985

ima odgovarajući broj 10, bajzik naredbu dodele vrednosti i podatak 1985.

Mašinski program ima potpuno istu strukturu. Program je sačinjen od linija mašinskih naradbi i svaka od njih je smeštena na odgovarajući način u posebnu memorijsku lokaciju. Adresa te memo-

rijske lokacije označava položaj naredba u programu. Svaka linija mašinskog programa se sastoji od naredbe i, eventualno, podatka.

Spektrumova centralna procesorska jedinica izvršava 686 različitih mašinskih naradbi. Naredba je sastavljena od kôda operacije (angl. instruction coda) i operanda – podatka koji naredba koristi. Kôd operacija kao i operand mogu imati dužine od 1 ili 2 bajta. Na taj način mašinska nareda može u meoriji računara da zauzima najviše 4 bajta.

Zbog velikog broja kodova naredbi uvedan ja simbolički mašinski jezik u kome su naredbe prikazane svojim mnemoničkim oznakama.

U cilju ilustracije do sada izloženog dat je uporedni prikaz bajzik programa i akvivalnetnog mašinskog programa.

program bejzik	mašinski program				
	adrase		mnemonika	kodovi	
	decimal	heks.		decimal	heks.
50 LET A=100	30000	7530	LD A,100	62,100	3E,64
100 LET HL=23609	30002	7532	LD HL,23609	33,57,92	21 39 5C
110 POKE HL,A	30005	7535	LD (HL),A	119	77

Na adrasama 30000 (7530h) nalazi so mašinska naradbe koja u registar A stavlja sadržaj 100 (64h). Registar A je osmobitna mamorijska lokecija u samom mikroprocesoru. Naredba se nalazi u dve memorijske lokacije. U prvoj je smešten kôd operacija (62), a u drugoj je operand 100. Mnamonička oznaka naredba je LD A,100 Na adresama 30002, 30003 i 30004 nalazi se trobajtna naredba koja u HL par ragistara stavlja sadržaj 23609 (5C39h). Kôd operacija 33 je dat u jednoj memorijskoj lokaciji, a u sledeća dva je data vrednost operanda 57 i 92. Treća naredba je jednobajtna i njen je kôd, 119, a u memorijskoj lokaciji 30005. Pomoću nje sa sadržaj registra A prebacuje u memorijsku lokaciju čija adresa ja u paru registara HL.

Prethodni primer ima za cilj da pokaže da se mašinski program sastoji od nize kodova koji su dati u obliku osmobitnih vrednosti (bajtova) u memorijskim lokacijama. Ti podaci predstavljaju kodove operacija (naredbi) i operanda.

Proces sinteza mašinskog programa se sastoji u pisanju naredbi na simboličkom mašinskom jeziku i prevodanju u odgovarajuće kodove. Dobijene kodove zatim trebe smestiti u odradeni deo RAM-

-a. Za kratke programa se proces prevođenja obavlja pomoću tabela kodova. Ti kodovi sa pomoću bejzik naradbe **POKE** mogu smestiti u memoriju.

Unošanje dužih programa na ovakav način je neprihvatljivo, pa se zbog toga i taj deo posla prepušta računaru. To sa postiže tako što sa koristi program specijalno napravljen za unošenje podataka u memoriju. Takav program se naziva punjač (LOADER).

Primeri mašinskog programa dati u ovoj knjizi prikazani su u simboličkom mašinskom jeziku sa odgovarajućim kodovima. Dat je jedan program koji omogućava unošenje heksadecimalnih kodova i može sa primeniti u narednim primerima (deo 4-8,str. 176).

Program za punjenje ubrzava unošenje kodova, ali ne omogućava njihovo dobijanje na osnovu mnemoničkih simbola. To obavlja program koji se zove asembler. Program sa može pisati na simboličkom mašinskom jeziku, a upotrebom asemblera se prevodi na mašinski jezik i smešta na odgovarajuće adresa u obluku koda (engl. object code). Ovakav način pisanja dužih mašinskih programa je jedini prihvatljiv način.

Upotraba asemblera olakšava pisanje programa jer on poseduje i program za unošenja naredbi i podataka (editor). U toku rada se signalizuju greške ispisivanja i formiranja asemblarskog formata. Velika prednost asemblara je mogućnost labeliranja, tj. pridodavanja imena adrasama memorijskih lokacija. Tako se željena lokacija u memoriji može adresirati navođenjem njena labele.

Asemblari namenjeni Spektrumu zauzimaju oko 5 – 10 Kbajta. Pra početka rada potrebno je njihovo učitavanje u memoriju, što oduzima vreme i slobodan memorijski prostor. Takođe, za rad sa asemblerom potrebno je upoznati sve asemblerske komande.

Još jedan veoma koristan program je disasembler. On na osnovu koda operacije prikazuje simbolički mašinski program. Takođe, poseduje i niz drugih korisnih mogućnosti, kao što je pregled i promena sadržaja memorije i registara samog procesora.

Mašinski program počinje da se izvršava uz pomoć bejzik naredbe **USR** a, gde je a adresa od koje treba da se izvrši mašinski program. Pun oblik naradbe koji se najčešće primenjuje je **RANDOMIZE USR** a (jer **USR** a ne može da se koristi samo).

•

4-2 BROJNI SISTEMI

Za razumevanje načina rada Spaktruma i programiranja na mašinskom jeziku potrebno je poznavanja binarnog i haksadecimalnog brojnog sistema kao i načina na koji se predstavljaju brojevi. Uz koncept brojnih sistema objasniće se apsolutna binarna forma, forma komplementa dvojka i petobajtna forma pradstavljanja brojeva. Prva dva forma su usvojene za sve osmobitne mikroračunara, a petobajtna forma je karaktaristika Spektruma.

BINARNI BROJEVI

Brojni sistam je zasnovan na osnovi 10 naziva sa decimalnim brojnim sistemom. Osnova deset znači da sa pri brojanju posle svakih daset jedinica vrši prenos u viši težinski razrad. U decimalnom sistemu težine pojedinih težinskih razreda su 1, 10, 100,... i ona pradstavljaju stapene osnova daset: 10^0, 10^1, 10^2,... Decimalna cifre su 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Broj 125 se u dacimalnom sistemu pradstavlja:

$1*10^2 + 2*10^1 + 5*10^0 = 125$

Binarni sistem je zasnovan na osnovi dva. To znači da se posla dve jadinice vrši pranos u viši težinski razred. Težine pojedinih razreda su stapeni osnove dva: 2^0, 2^1, 2^2,..., a to su 1, 2, 4, 8,... Jedine dve binarne cifra su 0 i 1. Uočava se pogodnost ovih cifara za predstavljanja dva stanja napona (0 kada nema napona i 1 kada ga ima).

Decimalni broj 125 se u binarnom sistemu predstavlja na sledeći način:

$1*64 + 1*32 + 1*16 + 1*8 + 1*4 + 0*2 + 1*1 = 1111101$

dec.	bin.	heks.
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8

9	1001	9
10	1010	A
11	1011	B
12	1111	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12

Tabela 4–1 Uporedni prikaz decimalnih, binarnih i heksadecimalnih brojeva

APSOLUTNA BINARNA FORMA

Sa osam bita mogu se pradstaviti binarni brojavi od ØØØØØØØØ do 11111111, što u decimalnom sistemu čini opseg od Ø do 255.

U konceptu apsolutna binarne forme svih 256 brojeva su calobrojne i pozitivne veličine. Zbog osmobitne fizička realizacije registara i memorije brojavi veći od 255, kao i negativni brojevi, predstavljaju se na sledeći način: U slučaju broja većeg od 255 koji moža da nastane sabiranjam dva broja, rezultat če se pojaviti kao da ja smanjen za 256. Tako će broj 275 u registru ili memoriji biti predstavljen kao broj 19. Nagativni brojevi će biti predstavljeni kao da im je dodato 256. Broj – 23 će se u mamoriji nalaziti kao broj 233.

Spektrum može da radi I sa šesnaestobitnim brojevima koji sa dobijaju od dva osmobitna. To su pozitivni celi brojevi u opsegu od Ø do 65535.

BINARNI BROJEVI U KOMPLEMENTU DVOJKE

. Koncept komplemanta dvojke uveden je sa ciljem predstavljanja negativnih brojava. Od osam bita, za označavanje predznaka uzet je bit najveće težine. Taj bit sa naziva bit znaka i za pozitivna brojave je Ø, a za negativna je 1. Sa preostalih sedam bita predstavljaju se 128 pozitivnih i 128 negativnih brojava.

decimalni brojevi	binarni brojevi u komplementu dvojke
127	Ø1111111
126	Ø1111110
2	ØØØØØØ1Ø
1	ØØØØØØØ1
Ø	ØØØØØØØØ

-1	11111111
-2	11111110
-127	10000001
-128	10000000

Tabela 4-2Uporedni prikaz decimalnih brojeva i binarnih brojeva u komplementu dvojke

Negativni brojevi se formiraju na sledeći način. Prvo se nađe komplement odgovarajućeg pozitivnog broja. To se postiže prevođenjem nule u jedinice i jedinica u nule. Dodavanjem jedinice formira se željeni broj izražen u komplementu dvojke

+7 00000111
11111000 komplement
+ 1
−7 11111001 komplement dvojke

Prvi bit sa leve strane je bit znake.

Na isti način se iz negativnih brojeva dobijaju pozitivni što je od praktičnog značaja pri programiranju na mašinskom jeziku.

Koncept komplementa dvojke je primenljiv i na šesnaestobitne brojeve pri čemu je opseg brojeva od -32768 do 32767

PETOBAJTNA FORMA PREDSTAVLJANJA BROJEVA

Radi povećanja opsega korišćenih brojeva i radi upotrebe realnih brojeva u Spektrumu se primenjuju dve petobajtne forme. Prva se odnosi na pozitivne i negativne cele brojeve, a druga na pozitivne i negetivne realne brojeve.

Za predstavljenje celih brojeva koristi se pet bajtova. Prvi bajt je uvek 0. Drugi bejt ja 0 ako je broj pozitivan, e 255 ako je negativan. Treći i četvrti bejt sadrže vrednost broja po konceptu šesnaestobitnog komplementa dvojke, s tim što svih 16 bita određuju vrednost broja. To omogućuje upotrebu brojeva iz opsege od -65535 do +65535. Peti bajt je uvek nula

Sledeći program omogućuje prikazivenje broje u petobejtnoj formi. Na osnovu sistemske promenljive VARS nelazi se programska promenljiva a **(INPUT a)** i ispisuje se pet bejtova kojima je ona predstavljena

```
10 INPUT a: PRINT "Broj:";a
20 LET b=PEEK 23627+256*PEEK 2
3628
30 FOR n=1 TO 5: PRINT n;"-";P
```

```
EEK (b+n): NEXT n: GO TO 10
```

Realni brojavi se predstavljaju drugom petobajtnom formom koja pokriva opseg brojeva od oko 2.94E–39 do oko 1.7E38 kako pozitivnih, tako i negativnih.

Realni broj A koji sa predstavlja na ovaj način prevodi se u oblik ±M*2↑E gda je M mantisa broja (0.5≤M<1), a E je eksponent tog broja. Eksponent se nalazi prama izrazu:

E=1+INT(LN(ABS A)/LN2),

a mantisa:

M=ABS A/2↑E

Prvi bajt petobajtne forme dobija vrednost zbira 128+E. Ukoliko je A pozitivan broj, drugi bajt dobija celobrojnu vrednost proizvoda broja 256 i mantise umanjenog za 128. Za negativne vrednosti A ne oduzima se broj 128. Decimalni ostatak dobijen nalaženjem prethodne vrednosti množi sa sa 256. Celobrojni deo je vrednost trećeg bajta, a od decimalnog ostatka se na isti način formira četvrti i peti bajt. Ako je poslednji decimalni ostatak veći od 0.5, vrednost petog bajta se povećava za 1.

Gornji program dopunjen programskom linijom 15:

```
15 LET a=a+1E-38
```

koristan je za razumevanje ovog načina predstavljanja brojeva. Linija 15 osigurava da se celi brojevi od –65535 do +65535 na prikazuju u petobajtnoj celobrojnoj formi.

Raalni brojevi predstavljeni na ovaj način nazivaju se brojevi sa pomičnim zarezom (engl. floating point).

HEKSADECIMALNI BROJEVI

Heksadecimalni brojni sistem je zasnovan na osnovi brojanja šesnaest. To znači da će se posla šesnaest jedinica izvršiti prenos u viši težinski razred. Težina pojadinih težinskih razreda su 1, 16, 256,... One predstavljaju stepene osnove šesnaest. Heksadacimalne cifre su 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E i F. U tabeli 4–1 dat je uporedni prikaz decimalnih binarnih i heksadecimalnih brojeva.

Broj 225 se u heksadecimalni sistem prevodi na sledeći način:

$$14*16^1 + 1*16^0 = 225$$

što daje haksadecimalni broj E1.

Oznakom # ispred broja ili h iza broja ukazuja se da se radi o heksadecimalnom broju.

Haksadecimalni brojevi sa najčašća primenjuju za kodove naredbi i obeležavanje memorijskih adresa.

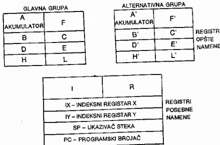
4–3 MIKROPROCESOR Z80

Centralna procesorska jedinica (CPU) je glavni dao računarskog sistema. Njegova uloga je da na osnovu naredbi izvršava željene operacije. U Spektrumu je centralna procesorska jedinica mikroprocesor Z80.

U njegovoj strukturi se uočavaju tri celine: registri, aritmetičko-logička jedinica i kontrolna jadinica sa registrom naredbi.

REGISTRI

Z80 sadrži 208 bita RAM memorije koji su dostupni programeru. Na slici 4–1 je prikazano kako ja ta memorija raspoređana u 18 osmobitnih i 4 šesnaastobitna ragistra.



Slika 4–1 Registri procesora Z80

A REGISTAR

Ovaj registar se naziva još i akumulator, jer se u njega smeštaju rezultati aritmetičkih i logičkih operacija. Predstavlja najčešće korišćeni registar u naredbama mikroprocesora.

F REGISTAR

F registar se posmatra kao grupa od 8 bita od kojih svaki ukazuje na neko stanje do koga se došlo u toku izvođenja operacije. Za programiranje su od značaja četiri bita F ragistra koji se nazivaju pokazateljima i to su: Z – pokazatelj nule (zero flag), C – pokazatelj pranosa (carry flag), S – pokazatelj znaka S (sign flag) i P/V – pokazatelj parnosti/prenošenja (parity/overflow flag). Ostala četiri bita interno koristi kontrolna jedinica i ne mogu se direktno upotrebiti.

HL, BC i DE REGISTRI

Svaka od dva grupe registara opšte namene sadrži po šest registara koji mogu biti upotrebljeni zasebno kao osmobitni ili u paru kao šesnaestobitni registri.

U mikroprocesoru Z80, HL registar par je najvažniji od tri registar para koji mogu da sadrže adresu. Neke naredbe su ostvarive samo uz njegovu primenu. HL ragistar par može takođe u sebi sadržati i šesnaestobitni broj sa kojim može učestvovati u aritmetičkim operacijama.

Svaki od registara opšte namene moža biti upotrebljen zasebno, s tim da je broj naredbi koji se može izvršiti sa njima manji nego sa A registrom.

ALTERNATIVNI REGISTRI

Izvršavanjem potrebnih naredbi, Z80 prelazi na rad sa grupom registara koja je do tada bila alternativna, a grupa koja je bila glavna postaje altarnativna. Unutar samog procesora ne postoji informacija koja grupa sa tranutno koristi.

Potpuno iskorišćavanje mogućnosti procesora pretpostavlja upotrabu alternativnih registara, pogotovo u slučajevima kada procesor prelazi na privremeno izvršavanje nekog drugog programa (npr. program prekida).

Jednostavnim prelaskom na altarnativnu grupu eliminiše sa procedura čuvanja sadržaja registara radi njihova kasnije upotrebe.

PC – PROGRAMSKI BROJAČ

Programski brojač (angl. program counter) je 16-bitni ragistar posebne namene koji sadrži adresu memorijske lokacija iz koje se prauzima kôd naredbe koja treba da se izvrši. Sadržaj PC registra se automatski povećava nakon što je njegov sadržaj postavljen na adresnu magistralu i pošto je procesor učitao kôd odgovarajuće naradbe. Naradbe skoka postavljaju nove vradnosti u PC registar. Te nove vrednosti su adrase od kojih se nastavlja izvršavanje programa.

SP – UKAZATELJ STEKA

Ukazatalj steka (angl. stack pointar) ja 16 bitni ragistar koji sadrži adresu početka dela RAM memorije koji se naziva stak. Stek služi za privremeno smeštanje podataka i adresa (npr. pri skoku u potprogram).

Izvršavanjem odgovarajućih mašinskih naredbi podaci sa iz jednog od para ragistara stavljaju na stek ili sa staka vraćaju u jedan od parova registara. U naredbi PUSH ilustruje se izvršavanje takve operacije

Stek je organizovan po principu "poslednji unutra – prvi napolje" (LIFO –last–in–first–out). Podatak koji se prvi stavlja na stek moža se sa staka vratiti samo kao poslednji. Odnosno, podatak koji je kao posladnji stavljen na stek moža biti vraćan sa steka samo kao prvi

Stak moža biti smešten bilo gde u RAM-u. U Spektrumu je njagov položaj određen sistamskom promanljivom RAMTOP i nalazi se ispod G0 SUB steka. Upotrebom staka skraćuje se dužina programa i povaćava njegova brzina izvršavanja.

IX I IY – INDEKSNI REGISTRI

Indeksni registri (engl. index ragisters) omogućavaju indeksirano adresiranje kod koga je adresa memorijske lokacije jednaka zbiru sadržaja indaks ragistra (bazna adresa) i operanda koji je navaden u naradbi. Ovo ja naročito pogodno za obradu tabela podataka

I – REGISTAR PREKIDA

Registar prekida (engl. interrupt register) omogućava jadan način dobijanja početne adrese programa za servisiranja prakida (o

prakidu više u naredbama kontrole procesora). Pri toma registar prekida ukazuje na viši bajt, a periferna jedinica na niži bajt adrese na kojoj se nalazi adresa programa prekida.

R – REGISTAR ZA OSVEŽAVANJE MEMORIJE

Upotraba dinamička RAM memorije zahteva da se 500 puta u sekundi vrši osvežavanje njenog sadržaja. Procas osvažavanja se obavlja automatski ne utičući na rad procesora. Sadržaj registra za osvežavanje (engl. refresh) sa koristi kao niži, a sadržaj I registra kao viši adresni bajt. Posle svake izvršana naredbe adresa se povećava za 1 i vrši se osvežavanje adresirane lokacije.

ALU – ARITMETIČKO LOGIČKA JEDINICA

ALU (engl. arithmetic and logic unit) je dao mikroprocesora koji omogučava da se nad podacima obave sledeće aritmatičke i logičke operacije.

sabiranje
oduzimanje
logičko I
logičko ILI
logičko isključivo ILI
poređenje
logička i aritmetička rotacija
levo i desno

povećanja za jadan
smanjenja za jedan
postavljanje bita (sat)
brisanje bita (reset)
tastiranje bita

KONTROLNA JEDINICA I REGISTAR NAREDBI

Kôd svaka naredbe se čita iz memorija i upisuje u registar naredbi mikroprocasora (angl. instruction register). Kontrolna jedinica (engl. CPU control) dekoduje kôd naredbe i na osnovu njega generiše potrabne unutrašnje i spoljašnje signale koji su potrebni da bi se željena oparacija obavila.

4–4 NAČINI ADRESIRANJA

Naredba mikroprocesora Z80 se odnose na podatke koji se nalaze u njegovim internim ragistrima, spoljnoj RAM mamoriji ili perifernim jedinicama. Način adresiranja pokazuje kako se vrši adresiranje željenog podatka. Ovde sa traba samo upoznati sa mo-

gućim načinima adresiranja, a njihovo potpuno razumevanje će uslediti tokom pregleda naredbi.

NEPOSREDNO ADRESIRANJE (immadiate)

Bajt koji sledi posle kôda operacije je operand (podatak) potraban za izvršenje naradbe

KOD OPERACIJE	1 ili 2 bajta
OPERAND	
b7 b0	

Primer ovakvog adresiranja je punjenje akumulatora konstantom.

PRODUŽENO NEPOSREDNO ADRESIRANJE (immediate extended)

Ovaj način je prošireni oblik neposrednog adresiranjá. Razlika je u tome što je operand dvobajtna veličina.

KOD OPERACIJE	1 ili 2 bajta
OPERAND	bajt manje težine
OPERAND	bajt veća težine

Primer je punjenje HL registar para 16-bitnom konstantom.

MODIFIKOVANO ADRESIRANJE NULTE STRANE (modified page zero addressing)

Poseban oblik naredbi pôziva (RST) se odnosi na 8 od 256 lokacija nulte strane koja se mogu adresirati jednim bajtom

KOD OPERACIJE	1 bajt

RELATIVNO ADRESIRANJE (relative addrassing)

Za ovo adresiranje se koristi jedan bajt koji sledi posle koda operacije. On ukazuje na udaljenost memorijske lokacije od koje ća se nastaviti izvršavanje programa. Relativno adresiranje se odnosi na naredbe skoka, koji se može ostvariti do 128 memorijskih lokacija unapred i do 127 memorijskih lokacija unazad.

KOD OPERACIJE	relativni skok
OPERAND	8 bita u komplementu dvojke

PRODUŽENO ADRESIRANJE (extended addressing)

Posle koda operacije slede dva bajta koji su adresa na koju se skaće ili adresa operanda koji će biti korišćen.

KOD OPERACIJE	1 ili 2 bajta
ADRESA	bajt manje težine
ADRESA	bajt veće težine

INDEKSIRANO ADRESIRANJE (indexed addressing)

Bajt koji sledi posle koda operacije sadrži podatak čija se vrednost dodaje sadržaju jednog od dva indeksna registra. Tako se dobija odgovarajuća adresa lokacije u memoriji.

KOD OPERACIJE	dva bajta koda
KOD OPERACIJE	operacije
UDALJENOST	vrednost koja se dodaje indeksnim registrima

Primer ovakve naredbe je punjenje akumulatora sadržajem memorijske lokacije (INDEKS REGISTAR + udaljenost).

REGISTARSKO ADRESIRANJE (register addressing)

Odnosi se na razmenu podataka između samih registara. Primer je punjenje registra A sadržajem registra B.

INDIREKTNO REGISTARSKO ADRESIRANJE (register indirect addressing)

Ovaj način adresiranja koristi 16-bitni registar par (na primer HL) za ukazivanje na određenu memorijsku lokaciju. Primer je punjenje akumulatora sadržajem memorijske lokacije adresirane sadržajem HL registar para.

IMPLICITNO ADRESIRANJE (implied addressing)

Pri ovom načinu adresiranja operandi se nalaze u jednom ili više registara mikroprocesora. Primer su naredbe aritmetičkih operacija gde se u akumulator uvek smešta rezultat operacije.

ADRESIRANJE BITA (biti addressing)

U naredbama mikroprocesora Z80 postoji grupa naredbi pomoću kojih se bitovi mogu postavljati na nulu ili jedinicu. Takođe se

može vršiti njihovo testiranje kao i bilo kog registra ili memorijske lokacije. To se ostvaruje kroz registarsko, indirektno registarsko i indirektno adresiranje.

4- 5 NAREDBE MIKROPROCESORA Z80

Mikroprocesor Z80 može izvršiti ukupno 158 osnovnih naredbi koje se mogu podeliti u 8 grupa:
1. naredbe punjenje i zemene (load and exchange)
2. eritmetičke i logičke naredbe (eritmetic and logic)
3. naredbe skoka, poziva i povratka (jump, call end return)
4. naredbe prebecivanja blokove i pretraživanja (block transfer and search)
5. neredbe rotacije i pomerenja (rotate and shift)
6. naredbe menipulecije bitovime (bit menipulation)
7. ulazno-izlazne naredbe (input/output)
8. neredbe kontrole procesora (CPU control)

1. NAREDBE PUNJENJA I ZAMENE

Naredbe osmobitnog punjenja

Naredbe punjenja prebacuju podatke interno između registara procesora, ili između njih i spoljne memorije. Primer ze interno prebacivanje je prebacivenje sadržaje registra B u registar C. Primer prebacivanja podataka između registara i spoljne memorije je prebacivanje sadržaja memorijske lokacije NN u registar A. Pri tome NN označava bilo koju memorijsku lokaciju od Ø do 65535.

Osgovne kerekteristike neredbi punjenja je da se sedržaj mesta sa koga se uzima podatak ne menja.

U tabeli 4-3 su date sve naredbe (sa heksadecimalnim kodovima) punjenje osmobitnim brojem. Tekode su naznačeni i tipovi adresirenja. U gornjem redu se nalazi registar ili memorijska lokacija sa koje se podatak uzima, a u prvoj koloni registar, ili memorijska lokacija, u koju se podatak stevlja. U preseku odgovarajućih kolona i redove nelazi se kôd operacije. Gornji primer prebacivenje sadržeje registra B u register C ime kôd 48h (četvrta kolona, treći red).

Mnemoničke oznaka naredbi punjenja uvek započinje oznekom LD, što je skraćenica od LOAD (napuniti). Nakon toga slede oznake meste koje se puni i mesta sa koga se podatak uzime. Oz-

ODREDIŠTE (DESTINATION)		POLAZNO MESTO (SOURCE): IMPLICITNO (IMPLIED)		REGISTARSKO (REGISTER)							INDIREKTNO REGISTARSKO (REG. INDIRECT)			INDEKSIRANO (INDEXED)		PROŠIRENO (EXT. ADDR.)	NEPOSREDNO (IMME.)
		I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	(nn)	n
REGISTARSKO (REGISTER)	A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
	B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
	C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
	D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
	E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
	H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
	L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
INDIREKTNO REGISTARSKO (REG. INDIRECT)	(HL)			77	70	71	72	73	74	75							36 n
	(BC)			02													
	(DE)			12													
INDEKSIRANO (INDEXED)	(IX+d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
	(IY+d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
PROŠIRENO (EXT. ADDR.)	(nn)			32 n n													
IMPLICITNO (IMPLIED)	I			ED 47													
	R			ED 4F													

Tabela 4–3 Naredbe 8-bitnog punjena

nake su odvojena zarezom. Naredba punjenja registra C sadržajem registra B označava se:

LD C,B

U naredbama punjanja se upotrebljavaju i zagrada. Na primer naredba.

LD (HL), E

označava operaciju u kojoj se sadržaj registra E ne prebacuje u registar par HL, već u memorijsku lokaciju čija je adrsaa sadržana

8·

u HL. Isto tako naredba **LD A,(nn)** označava se će se akumulator napuniti, ne brojem nn, već sadržajam memorijska lokacije čija je adrasa nn.

Sagledavejući tabalu 8-bitnog punjenja uočavaju se najveće mogućnosti akumulatora. On može razmenjivati svoj sadržaj sa

- svim registrima opšte namene
- mamorijskim lokacijama na koje svojim sadržajem ukazuju ragistar parovi opšte namene
- memorijskim lokacijama na koje ukazuju indeksni registri
- memorijskom lokacijom koja je određena 16-bitnom konstantom
- I i R registrom

Akumulator ja takođe moguće direktno napuniti 8-bitnom konstantom.

Registri opšte namene imaju nešto manje mogućnosti. Mogu razmenjivati podatke sa ostalim registrima opšte namene i sa mamorijskim lokacijama na koje ukazuju registar par HL ili indeksni registri. Takođe se mogu diraktno napuniti i željenom konstantom.

Preostale dve naredbe u kojima ne učestvuje ni akumulator ni registri opšte namene su punjenje konstantom onih mamorijskih lokacija na koja ukazuju HL ili indeksni ragistri.

Data tabela omogućava jasno sagladavanje svih mogućih obilka naredbi punjenja 8-bitnim brojem uz odgovarajuće kodova procesora Z80. Naredbe su dužne od 1 do 4 bajta. Kôd oparacije je dat haksadecimalno. Slovom N je označana 8-bitna konstanta kojom se mogu puniti akumulator, registri opšta namena i memorijske lokacija adresirane HL i indeksnim registrom.

Oznakom d u naredbama sa indeksnim registrima obeležana je udaljenost memorijske lokacije koja učestvuje u naredbi od bazna lokacije označene indeksnim registrom. Osmobitni broj d ja u komplementu dvojke što omogućuje adresiranje memorijskih lokacija udaljanih do +127, odnosno −128 lokacija od bazne adresa.

Naredbe punjenja se u mamorijl nalaze u sledećim oblicima:

- jednobajtne naredbe (zauzimaju 1 bajt memorije) sadrže samo kôd operacije
- dvobajtne naradbe, zauzimaju 2 bajta memorije u sladećam poretku

adresa A 1Eh kôd operacije
adresa A+1 63h operand (osmobitni)

Primer se odnosi ne naredbu punjenja registra E sa vrednošću 63h.

– trobajtna naredbe punjenja u memoriji se nalaze u sledeće dve forme. Prva se odnosi na naredbe koje korista indeksne registre. Od tri bajta, prva dva su kôd same operacije, dok je treći bajt operand udaljenosti d. Na primer, punjenje registra E sa sadržajam memorijske lokacije koja se nalazi na adresi IX + 22 postiže se naredbom **LD E, (IX+22)** koja se u memoriji nalazi ovako:

adresa A DDh
adresa A+1 5Eh kôd operacije
adresa A+2 22h operand

udaljenosti

Dve naredbe, koje prebacuju podatka iz akumulatora u memorijsku lokaciju, odnosno iz memorijske lokacija u akumulator, takođe su trobajtna. Na primer, punjanje akumulatora sistemskom promenljivom LAST K piše se **LD A, (5C08)**, a u memoriji je ova naredba sadržana u obliku:

adresa A 3Ah kôd operacije
adresa A+1 08h niži bajt adrese
adresa A+2 5Ch viši bajt adrese operand

– četvorobajtne naredbe su naredbe punjenja memorijske lokacije označene indeksnim registrima i odgovarajućom udaljenošću d. Naredba **LD (IY-7),15** u memoriji je u sledećem obliku:

adrasa A FDh
adrasa A+1 36H kôd operacije
adrasa A+2 F9h udaljenost

(−7 u komplementu dvojka)

adrasa A+3 15h vrednost kojom se puni

Primeri koji ilustruju mašinska naredbe dati su kasnije. Osmobitno punjenje je prikazano u primarima 1, 2 i 3.

Naredba šesnaestobitnog punjanja

Tabela 4–4 prikazuja naradba (i kodova) za punjenje šesnaestobitnim brojem. Ova tabela ja veoma slična prethodnoj. U gornjem radu sa nalaze registar parovi, par memorijskih lokacija i 16-bitna konstanta. Oni predstavljaju mesto sa koga se uzima 16-bitni broj. U prvoj koloni su registar parovi i memorijske lokacije u koje se prebacuje 16-bitna vrednost. Presak pripadajućih kolona i redova daje kodove i naredbe. Na primer, punjanja ragistar para HL 16-bitnom konstantom obeležava se:

LD HL,nn

Kôd naredba je dat u preseku četvrtog reda i osme kolone, e njen oblik u memoriji je:

Adresa A 21 kôd operacije
Adresa A+1 n niži bajt ide u registar L
Adresa A+2 n viši bajt ide u registar H operand

		POLAZNO MESTO (SOURCE): REGISTARSKO (REGISTER) AF	BC	DE	HL	SP	IX	IY	PRODUŽENO NEPOSREDNO (IMM. EXT.) nn	PRODUŽENO ADRESIRANJE (EXT. ADDR.) (nn)	INDIREKTNO REGISTARSKO (REG. INDIR.) (SP)
ODREDIŠTE (DESTINATION) REGISTARSKO (REGISTER)	AF										F1
	BC								01 n n	ED 4B n n	C1
	DE								11 n n	ED 5B n n	D1
	HL								21 n n	2A n n	E1
	SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
	IX								DD 21 n n	DD 2A n n	DD E1
	IY								FD 21 n n	FD 2A n n	FD E1
ADRESIRANJE (EXT. ADDR.)	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
NAREDBE PUSH (REG. INDIR.)	(SP)	F5	C5	D5	E5		DD E5	FD E5			NAREDBE POP

Tabela 4-4 Naredbe 16-bitnog punjena

U bloku 16-bitnog punjenja, naredbe koje se razlikuju od ostalih su: **PUSH** (stavi na stek) i **POP** (vrati sa steka). Punjenje memorijske lokacije na koju ukazuje registar SP, na primer, sa sadržajem BC registra, ne označava se LD (SP), BC. Jedino dozvoljano i ispravno označavanje je

PUSH BC

Izvršavanjem ove naredbe dešava se sledeće: Vrednost registra SP se smanjuje za jedan, ukazujući na novu memorijsku lokaciju

Izvrši se punjenja te lokacije sadržajem registra B. Vrednost registra SP se ponovo smanjuje za jedan, ukazujući na sledeću lokaciju koja sa puni sadržajem C registra. To se može prikazati na sledeći način:

Pre izvršenja **PUSH BC**		Posle izvršenja **PUSH BC**	
Adrasa A	[ ] ← SP ukazuje na adresu A	Adrasa A	[B]
Adresa A-1	[ ]	Adresa A-1	[C]
Adresa A-2	[ ]	Adresa A-2	[ ] ← SP ukazuje na adresu A-2

Naredba **POP** ima suprotan efekat od naredbe **PUSH**. Naredba **POP BC** povećava vrednost SP za jedan, prebacuje sadržaj ukazana mamorijske lokacija u C ragistar, ponovo povećava SP i prebacuja sadržaj nove lokacije u B ragistar.

Za **PUSH** i **POP** je značajno da sa izvršavaju nad parom registara. Pri tome se pri izvršavanju naredbe **PUSH** na stek uvek prvo stavlja viši bajt, a pri naredbi **POP** u registar par uvak se prvo vraća niži bajt. To znači:

PUSH BC stavi B pa C na stak
POP HL vrati u L pa u H sa staka

Kada se na stek stave dva bajta iz nekog para registara, oni mogu biti vraćeni u drugi par registara. Sledeći mašinski program od dve naredbe izvršio je kopiranja sadržaja AF registara u BC registar par.

PUSH AF stavi na stak sadržaje AF registara
POP BC vrati sa staka u BC registre
Primari 1, 3 i 4 prikazuju naradba ove grupe.

Naredba zamene

Ova naradba vrše zamenu sadržaja ragistara. Sadržaj jednog registra prebacuje se u drugi, a sadržaj drugog ragistra u prvi. Te naredba su 16-bitnog tipa, odnosno primenljiva su samo na registre u paru i memorijske lokacije u paru.

Tabela 4–5 ilustruje naradbe zamane Z80 procesora. Heksadecimalni kôd 08 dozvoljava izbor jednog od dva para A i F registara. Kôd D9 daje mogućnost izbora jedne od dve grupe po 6 registara opšte namene. Kôd EB odgovara veoma često primenjivanoj naradbi zamene sadržaja registar parova DE i HL. Kôd E3 ja kôd na-

		IMPLICITNO ADRESIRANJE				
		AF'	BC',DE' & HL'	HL	IX	IY
IMPLICITNO (IMPLIED)	AF	08				
	BC, DE & HL		D9			
	DE			EB		
(REG INDIR)	(SP)			E3	DD E3	FD E3

Tabela 4–5 Naredbe zamene

redbe zamene sadržaja registar para HL i memorijske lokacije na koju ukazuju SP.

Kodovi svih ovih naredbi su dužine 1 bajt. Preostale dva naredbe zamene dužine od po dva bajta zamenjuju sadržaje indeksnih registara sa memorijskim lokacijama adresiranih registrom SP.

Mnamoničko označavanje naredbi je:

kôd	mnemonika
08	**EX AF, A'F'**
D9	**EXX**
EB	**EX DE,HL**
E3	**EX(SP),HL**
DD E3	**EX(SP),IX**
FD E3	**EX(SP),IY**

Oznaka **EX** (od engleske reči exchange – zameniti), praćena je parovima čije se vrednosti zamenjuju. Izuzetak je zamena seta registara opšte namene koja ima rezervisanu oznaku **EXX**.

Način upotrebe naredbi ove grupe ja dat primerom 3.

2 ARITMETIČKE I LOGIČKE NAREDBE

Pomoću naredbi ove grupe obavijaju se aritmatičke operacije sabiranja, oduzimanja, povećanja i smanjenja za jedan, kao i logičke operacije sabiranja, množenja i poređenja

8-bitne aritmetičke i logičke naredbe

Sve 8-bitne aritmetičke i logičke operacije su date u tabeli 4-6. U izvršavanju svih naredbi sem naredbi **INC** i **DEC** učestvuje akumulator. Operacije se izvode nad sadržajem akumulatora ili između sadržaja akumulatora i sadržaja registra opšte namene ili memorijske lokacije adresirane HL, IX ili IY registrom. Aritmetičko-logičke operacije se takođe mogu izvršavati između akumulatora i 8-bitne konstante.

POLAZNO MESTO (SOURCE)

	REGISTARSKO ADRESIRANJE (REGISTER ADDRESSING)							(REG INDIR.)	INDEKSIRANO (INDEXED)		NEPOSREDNO (IMME)
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
ADD SABIRANJE	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADC SABIRANJE SA PRENOSOM	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUB ODUZIMANJE	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SBC ODUZIMANJE SA PRENOSOM	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
I AND	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
ILI OR	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
ISK. ILI XOR	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
POREĐENJE CP	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
POVEĆANJE INC	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
SMANJENJE DEC	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

Tabela 4-6 8-bitne aritmetičke i logičke operacije

Rezultat operacije se smešta u akumulator, a izuzetak je naredba poređenja **CP** koja ostavlja sadržaj akumulatora nepromenjenim.

Svaka od naradbi utiče na F registar, menjajući vrednost odgovarajućih bitova u zavisnosti od rezultata oparacije.

ADD (add) nareba vrši sabiranje sadržaja akumulatora i sadržaja navedenog polaznog mesta 8-bitnog podatka. Rezultet se smešta u A registar. Na primer, naredba sabiranja sadržaja A i B registara daje rezultat u A registru, dok B registar ostaja nepromenjen.

Pre operacije	A	FAh
	B	02h
Posle operacije	A	FCh
	B	02h

U ovom slučaju će pokazatalj prenosa (bit C ragistra F) biti postavljen na nulu, jer rezultat ne izlazi iz opsega od 00 do FF. Pokazatelj nule će takođe biti na nuli, jer rezultat nije nula.

Pre operacije	A	FAh
	B	09h
Posle operacije	A	03h
	B	09h

Bit C će u ovom primeru biti postavljan na jedinicu, a bit Z ća ostati na nuli.

ADC (add with carry) naradba sa razlikuje od prathodne po tome što sa zbir sadržaja akumulatora i 8-bitnog podatka sabira sa vrednošću C bita.

Naredba sabiranja akumulatora sa sadržajem D registra i C bitom označava sa:

ADC A,D

Pre operacije	A	E7h
	D	03h
	C	1h
Posla operacija	A	EBh
	D	03h
	C	0h

U slučaju da je C bit pra izvršenja operacija bio jednak nuli, u akumulatoru bi rezultat bio EA. Kada ja zbir sadržaja registara veći od FF događa se sledeće:

Pra operacije	A	E7h
	D	1Ah
	C	1h

Posle operacije	A	Ø2h
	D	1Ah
	C	1h

Pre izvršenja operacije C bit je mogao biti Ø. Teda bi rezultat u akumulatoru bio Ø1, a C bit bi se postavio na jedinicu.

SUB (subtrect) naredba omogućuje (u apsolutnoj binarnoj eritmetici) oduzimanje sedržeja naznačenih polaznih meste od sadržaja akumulatora.

Mnemonika ovih naredbi se vidi u primeru oduzimanja sadržaja registra L od A.

SUB L

Bit prenosa se postavlja na jedinicu ako je početna vrednost akumulatora manja od umanjioca, u ovom slučaju od vrednosti L registra. Za sadržaj A registra, koji je veći ili jednak umanjiocu, bit prenose se postavlja na nulu. Sledeći primer to ilustruje.

Pre operacije	A	2Fh
	L	1Ah
Posle operacije	A	15h
	L	1Ah
	C	Ø

U drugom primeru je vrednost sadržaja L registra veća od sadržaja akumulatora.

Pre operacije	A	2Fh
	L	4Ch
Posle operacije	A	E3h
	L	4Ch
	C	1h

SBC (subtract with carry). Ova naredba daje iste rezultate kao i **SUB** u slučaju da je pokazatelj prenosa C postavljen na nulu. C pokazatelj može biti jednak jedinici i tada se vrši njegovo dodatno oduzimanje od dobijenog rezultata.

Naredba je od koristi pri izračunavanjima u kojima je potrebna veća tačnost. Mnemonika ovih neredbi je prikazana primerom u kome se od sadržaje akumualtora oduzima sadržaj registre L.

SBC A,L

AND (and). Naredbom **AND** sa obavlja operacija logičkog množenja (I) između sadržaja akumualtora i drugog podatka. Operacija se izvodi bit po bit. Drugi podatak može da bude u bilo kojem registru ili memorijskoj lokaciji adresiranoj registrom.

U sledećem primeru sa operacija izvodi između sadržaja akumualtora i sadržaja memorijske lokacije adresirana HL registar parom

AND (HL)

akumulator	11010010
memorija	10000111
akumulator posle operacija	10000010

Kao i u ostalim aritmetičkim naredbama rezultat je smešten u akumulatoru.

Z i S bit se postavljaju na vrednosti koje zavisa od rezultata operacije, dok se C bit postavlja na nulu.

U prethodnom primeru se vidi da su neke jedinice u akumulatoru posle izvršenja operacije prebačena u nule. Taj postupak se obično naziva maskiranje i izvodi se drugim članom koji učestvuja u operaciji.

Da bi se C bit postavio na nulu potrebno je izvršiti **AND** naredbu između akumualtora i njagovog sadržaja. Tada se sadržaj akumulatora ne menja. Mnemonika ove naredbe je

AND A

OR (or).

Naredbom **OR** se obavlja operacija logičkog sabiranja (ILI) između sadržaja akumualtora i drugog podatka

OR N

akumulator	11010010
konstanta	10000111
akumulator posle operacije	11010111

U ovom primeru se obavlja operacija logičkog sabiranja između akumulatora i konstante date u binarnom obliku. Rezultat se

smešta u akumultor, a za bite pokazatelja stanja važi isto što i za prethodnu naredbu.

Naredbom OR se omogućava postavljanje željenih bita akumualtora na jedinicu.

XOR (exclusive or). Logička operacija isključivog sabiranja dve binarne cifre daje rezultat jedinicu ako je samo jedna, a ne obe cifre jednaka jedinici.

XOR B

akumulator	11010010
registar B	10000111
akumulator posla operacije	01010101

U ovom primeru je izvršena naredba **XOR** između sadržaja akumulatora i B registra. Za pokazatelje stanja važi isto što je i ranije izloženo za naredbe **AND** i **OR**.

Naredba **XOR A** se često koristi za postavljanje sadržaja akumualtora na nulu jer je to jednobajtna naredba u odnosu na **LD A,0** koja je dvobajtna naredba.

CP (compare). Ovo je jedna od vrlo često korišćenih naredbi koja vrši poređenje sadržaja A registra i podatka koji se nalazi na jednom od mesta datih u tabeli 4–6. Naredba poređenja obavlja oduzimanje dva binarna broja, s tim što se sadržaj registra ne menjaju. Jedino se može promeniti F registar u zavisnosti od rezultata operacije.

Pra operacije	A	2F
	L	1A
Posle operacije	A	2F
	L	1A
	C	0

INC (increment). Naredbom **INC** se vrednost sadržaja registra i memorijskih lokacija naznačenih u tabeli povećava za 1. Mnemonika ove naredbe u slučaju E registra je

INC E

DEC (decrement). Za ovu naredbu važi sve isto što i za naredbu **INC**, sem što se vrednost sadržaja registara ili mamorijskih lokacija smanjuje za 1.

DEC (HL)

Vrednost sadržaja memorijske lokacije adresirane HL registar parom se smanjuje za 1.

Aritmetičke naredbe opšte namene

Za procesor Z80 postoje i pet naredbi opšte namene koje se odnosa na akumulator i pokazatelj prenosa.

Komplementiranje akumulatora	'CPL'	2F
Dobijanje komplemente dvojke	'NEG'	ED 44
Komplementiranje bita prenosa	'CCF'	3F
Postavljanje na 1 bita prenosa	'SCF'	37
Prevođenje iz binernog u BCD oblik sadržaja akumulatora	'DAA'	27

Tabela 4–7 Aritmetičke naredbe opšte namene

CPL (complament). Izvršavanjem ove naredbe sadržaj akumulatora se zamenjuje svojim komplementom. Komplement binarnog broja nastaje prevođenjem jedinica u nule i nula u jedinice. O upotrebi komplementa je bilo više rečeno u 4.2.

NEG (negate accumulator). Ovom naradbom sa formira komplement dvojke broja sadržanog u akumulatoru. Efekat njena upotrebe je isti kao da je pvro izvršena naredba **CPL**, a zatim **INC A.**

CCF (complement carry flag). Promana vrednosti bita pren osa, C bita, obavija sa naradbom CCF.

SCF (set carry flag). Postavljanje C bita na jedinicu,, bez obzira na njegovo prethodno stanje, obavija se naredbom **SCF**. Postavljanje C bita na nulu postiža se obično naredbom **AND A.**

DAA (decimal adjust accumulator). Binarno kodovani decimalni broj (BCD) je broj u decimalnom sistemu čija je svaka cifra pradstavljena u binarnom obliku sa četiri bita. Decimalni broj 16 sa u BCD obliku piše kao 00010110, gde prva grupa od čatiri bita daje

broj 1, a druga grupa broj 6. Naredbom DAA se vrši prevođenje binarne vrednosti u BCD oblik.

16-Bitne aritmetičke naredbe

Naredbe ove grupe se obavljaju nad sadržajem 16-bitnih registara. Registri koji učestvuju u izvršavanju naredbi dati su u prvom redu i prvoj koloni sledeće tabele. Kao i u prethodno priloženim tabelama, rezultat operacije se smešta u registre prve kolone.

ODREDIŠTE (DESTINATION)		POLAZNO MESTO (SOURCE) BC	DE	HL	SP	IX	IY
'ADD' SABIRANJE	HL	09	19	29	39		
	IX	DD 09	DD 19		DD 39	DD 29	
	IY	FD 09	FD 19		FD 39		FD 29
SABIRANJE SA PRENOSOM 'ADC' I POSTAVLJANJE POKAZATELJA	HL	ED 4A	ED 5A	ED 6A	ED 7A		
'SBC' ODUZIMANJE SA PRENOSOM I POSTAVLJANJE POKAZATELJA	HL	ED 42	ED 52	ED 62	ED 72		
POVEĆANJE 'INC'		03	13	23	33	DD 23	FD 23
SMANJENJE 'DEC'		0B	1B	2B	3B	DD 2B	FD 2B

Tabela 4–8 16-bitne aritmetičke naredbe

ADD (addition). Ovom naredbom se obavlja sabiranje sadržaja dva šesnaestobitna registra. Sabiranje sadržaja IX sa sadržajem SP registra, pri čemu će rezultat biti smešten u IX registar, obavlja se naredbom

ADD IX,SP

ADC (add with carry and set flags). Sabiranje sadržaja dva šesnaestobitna registra postiže se naredbom ADC, s tim što se i

prethodna vrednsot pokazatelja prenosa C pridodaje zbiru. Rezultat se smešta u HL registar, a biti F registra se postavljaju u zavisnosti od rezultata.

Mnemonička oznaka šesnaestobitnog sabiranja, sa prenosom, između HL i BC registara je

ADC HL,BC

SBC (subtract with carry and set flags). Izvršavanjem ove naredbe se postiže oduzimanje šesnaestobitnih vrednosti registara pri čemu se uzima u obzir pokazatelj prenosa. Za oduzimanje sadržaja DE registar para od sadržaja HL registar para primenjuje se naredba sa mnemoničkom oznakom

SBC HL,DE

Rezultat se smešta u HL registar par, a pokazatelji stanja se postavljaju na vrednosti koje zavise od rezultata operacije.

INC (increment). Povećanje vrednosti sadržaja registar parova BC, DE, HL, SP, IX i IY za 1 ostvaruje se isto kao i u slučaju osmobitne aritmetike. Stanje F registra se ne menja.

Povećanje sadržaja para registara BC za jedan se postiže naredbom

INC BC

DEC (decrement). Smanjenje vrednosti sadržaja registar para za jedan postiže se naredbom **DEC**, pri čemu se stanje F registra ne menja.

Smanjenje sadržaja registra IX za jedan postiže se naredbom

DEC IX

3. NAREDBE SKOKA, POZIVA I POVRATKA

Naredbe ove grupe su posle naredbi punjenja najčešće primenjivane u mašinskim programima. Odnose se na promenu toka izvršavanja programa tako da se može reći da odgovaraju bejzik naredbama **GO TO, GO SUB** i **RETURN.**

Naredbe skoka (JUMP)

U toku izvršavanja programa, sadržaj programskog brojača PC se povećava izvršavanjem svake naredbe. Pri tome njegov sadržaj adresira memorijsku lokaciju u kojoj je smeštena sledeća naredba koja treba da se izvrši. Izvršavanjem naredbe skoka, programski brojač se postavlja, ne na sledeću, već na novu adresu od koje će se nastaviti dalje izvršavanje programa.

			USLOV (CONDITION)									
			BEZUSLOVNO (UN COND.)	IMA PRENOSA (CARRY)	NEMA PRENOSA (NON CARRY)	NULA (ZERO)	NIJE NULA (NON ZERO)	PARNO (PARITY EVEN)	NEPARNO (PARITY ODD)	NEGATIVNO (SIGN NEG.)	POZITIVNO (SIGN POS.)	REG B≠0
SKOK (JUMP) 'JP'	PRODUZENO NEPOSREDNO (IMMED EXT)	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
SKOK (JUMP) 'JP'	INDIREKTNO	(HL)	E9									
SKOK (JUMP) 'JP'	REGISTARSKO (REG. INDIR.)	(IX)	DD E9									
SKOK (JUMP) 'JP'		(IY)	FD E9									
SKOK (JUMP) 'JR'	RELATIVNO (RELATIVE)	PC+e	18 e−2	38 e−2	30 e−2	28 e−2	20 e−2					
SMANJUJE B SKOK AKO B≠0 DJNZ	RELATIVNO (RELATIV.)	PC+e										10 e−2
POZIV 'CALL'	PRODUZENO NEPOSREDNO (IMMED EXT)	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
POVRATAK (RETURN) RET	(REGISTER INDIR.)	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
POVRATAK IZ PREKIDA (RETURN FROM INT) RETI	(REG. INDIR.)	(SP) (SP+1)	ED 4D									
POVRATAK IZ NEMASKIRAJUĆEG PREKIDA (RETURN FROM NON MASKABLE INT) RETN	(REG. INDIR.)	(SP) (SP+1)	ED 45									

Tabela 4–9 Naredbe skoka poziva i povratka

U prvih šest redova tabele 4–9 prikazane su naredbe skoka. Da bi moglo da dode do izvođenja skoka, potrebno je da su zadovoljeni odrađeni uslovi. Ukoliko uslov nije ispunjen, ne dolazi do skoka i program nastavlja sa izvršavanjem od sledeće naredbe. Uslovi za obavljanje skoka dati su u prvom redu tabele i u zavisnosti od njih su mogući sledeći programski skokovi:

- bezusiovni skok
- skok uz uslov da ja pokazatelj pranosa (C bit) jednak 1
- skok uz uslov da je pokazatelj prenosa (C bit) jednak 0
- skok uz uslov da je pokazatelj nule (Z bit) jednak 1
- skok uz uslov da je pokazatelj nule (Z bit) jednak 0
- skok uz uslov da je pokazatelj parnosti (P bit) jednak 1
- skok uz uslov da je pokazatelj parnosti (P bit) jednak 0
- skok uz uslov da ja pokazatelj znaka (S bit) jednak 1
- skok uz uslov da je pokazatelj znaka (S bit) jednak 0
- skok uz uslov da je B registar različit od 0

Postojanje ovako velikog izbora uslova ze obavljanje skoka bitno olakševa pisenje mašinskih programa. Sem bezuslovnog skoka i skoka uz uslov da je B registar različit od nule, ostali programski skokovi su određeni stanjem pojedinih bita F registra. Značajno je da se prilikom skoka ne menja sadržaj F registra.

Postoje dve vrste skokova i to su:

– apsolutni
– relativni.

Naredbe apsolutnog skoke postavljeju u programski brojač adrese iz opsege od 0 do 65535. To znači da se može ostvariti skok na bilo koju lokaciju u memoriji od 64 Kbejta, odakle će se nastaviti sa izvršavenjem programa.

U drugom redu tabele, označenom sa **JP nn**, prikazani su kodovi naredbi neposrednog apsolutnog skoke na memorijsku lokaciju nn u zevisnosti od uslova.

Mnemoničke oznake naredbi za pojedine skokove su sledeće:

JP nn	– bezuslovni skok na adresu nn
JP C,nn	– skok uz uslov da je C bit jednak 1
JP NC,nn	– skok uz uslov de je C bit jednak 0
JP Z,nn	– skok uz uslov de je Z bit jednak 1
JP NZ,nn	– skok uz uslov da je Z bit jednak 0
JP PE,nn	– skok uz uslov da je P bit jednak 1
JP PO,nn	– skok uz uslov de je P bit jednak 0
JP M,nn	– skok uz uslov de je S bit jednek 1
JP P,nn	– skok uz uslov de je S bit jednak 0.

Neredbe apsolutnog skoka u memoriji zauzimaju tri bajta. U prvom bejtu je kôd operacije, a u druge dva je operand koji adresira memorijsku lokaciju ne koju se vrši skok.

U sledećem primeru je prikazane naredba za apsolutni skok na adresu 12ACh uz uslov da je Z bit jednak jedinici.

JP Z,12AC

A	23	kod operacije
A+1	AC	adresa sledeće neredbe
A+2	12	

Kodovi ze bezuslovni skok na adrese koje su sadržane u HL, IX i IY registar perovime prikazane su u drugom, trećem i četvrtom

delu tabele. Uočava se da naredba skoke na adresu sadržanu u HL registar paru zauzima jedan bajt memorije, dok ostale dve naredbe zauzimaju po dva bajta. Odgovarajuće mnemoničke oznake ovih naredbi su sledeće:

JP (HL) skok na memorijsku lokaciju adresiranu HL registar parom
JP (IX) skok na memorijsku lokaciju adresiranu IX registar parom
JP (IY) skok na memorijsku lokaciju adresiranu IY registar parom.

Naredbe relativnog skoka postavljaju adrese u programski brojač koje su do 127 lokacija veće ili do 128 lokacija manje od bazne adrese sa koje se obavlja skok. To su vrlo praktične naredbe koje omogućavaju kratke skokove. U memoriji ove naredbe zauzimaju dva bajta. U prvom bajtu je kôd operacije, a u drugom je operand koji predstavlja udaljenost memorijske lokacije sa koje se nastavlja dalje izvršavanje programa od bazna adrese. Operand je u formi komplementa dvojke.

Kodovi naredbi relativnog skoka su prikazani u petom redu tabele. Uočava se postojanje relativnih skokova koji mogu nastupiti u zavisnosti od svih uslova datih tabelom.

Mnemoničke oznake naredbi relativnog skoka su sledeće:

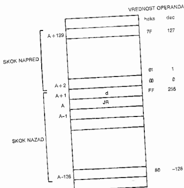
JR d	relativni bezuslovni skok
JR C,d	skok uz uslov da je C bit jednak 1
JR NC,d	skok uz uslov da je C bit jednak 0
JR Z,d	skok uz uslov da je Z bit jednak 1
JR NZ,d	skok uz uslov da je Z bit jednak 0

Na slici 4-2 je simbolično predstavljen deo memorije u okviru koga se obavlja relativni skok za različite vrednosti 8-bitne konstante d.

Pri skoku unapred broj d se može smatrati brojem preskočenih lokacija, dok se u skoku unazad malo složenije predstavlja

Najčešće korišćena naredba relativnog skoka je data mnemoničkom oznakom

DJNZ d



Slika 4–2 Adrese pri naredbama relativnog skoka

Ova naredba zauzima dva bajta memorije. Kada u toku izvršavanja program naiđe na naredbu skoka **DJNZ**, a da je pri tome sadržaj registra B različit od 0, obaviće se skok na lokaciju čija je udaljenost određena operandom d. Neposredno pre skoka se vrši umanjivanje vrednosti sadržaja B registra za 1. Ponovnim nailaskom na ovu naredbu ceo proces skoka se ponavlja, sve dok se vrednost sadržaja B registra ne izjednači sa nulom. Uočava se sličnost ove naredbe sa bejzik **FOR NEXT** petljom sa korakom – 1. Treba obratiti pažnju da se umanjivanje za 1 obavlja pre izvršavanja skoka, što znači da za vrednost sadržaja B registra jednakom 1 neće doći do skoka.

Time su prikazani svi oblici naredbe skoka a primeri 6 i 7 ilustruju njihovu primenu.

Naredbe skoka na potprogram (CALL)

Naredbe poziva su posebna vrsta naredbi skoka. Razlika je samo u tome što se u ovom slučaju, neposredno pre skoka, automatski stavlja sadržaj programskog brojača na stek. Time je nakon izvršenog skoka, negde u programu, sačuvana informacija o lokaciji sa koje je usledio skok. Vraćanjem vrednosti sa steka direktno u programski brojač, program nastavlja sa izvršavanjem od mesta sa koga je izvršen skok. Očigledno je da ove naredbe odgovaraju bejzik **GO SUB** naredbi.

CALL naredba je trobajtna u istom obliku kao i **JP** naredba. U prvom bajtu je sadržan kôd operacije, dok je u drugom i trećem apsolutna adresa mašinskog potprograma.

ADRESA POZIVA (CALL ADDRESS)	KÔD NAREDBE (OP CODE)	
0000H	C7	'RST 0'
0008H	CF	'RST 8'
0010H	D7	'RST 16'
0018H	DF	'RST 24'
0020H	E7	'RST 32'
0028H	EF	'RST 40'
0030H	F7	'RST 48'
0038H	FF	'RST 56'

Tabela 4–10 Naredbe restarta

Težina trećeg bajta je 256 puta veća od težine drugog.

U sedmom redu tabele su prikazani kodovi i uslovi skoka na mašinski potprogram. Mogući uslovi su potpuno isti kao i u slučaju JP naredbi tako da je i odgovarajuće mnemonika iste, osim što se umesto potprograma **JP** upotrebljava oznaka **CALL** (poziv). Tako se bezuslovni poziv potporogrema na adresi nn označeve sa:

CALL nn

U slučaju poziva uz uslov da je pokazatelj prenosa, C bit, jednak 1 monemonika je data u obliku

CALL C, nn

Ostale mnemoničke oznake **CALL** naredbi se dobijaju anelogno iz oznaka **JP** naredbi.

Primeri 8 i 15 se odnose na prikaz ove nerdbe.

Restert naredbe (Restart) su bezuslovne CALL naredbe na unapred dodređene edrese. U procesoru Z80 postoji ukupno 8 restart naredbi datih tabelom 4–10 u kojoj su prikazane adrese na koje se pozivaju, kodovi operacije i mnemoničke oznake. U mnemoničkim oznakama brojevi su decimalni. Uočava se da su restart naredbe u odnosu na **CALL** naredbe, dužine samo 1 bajt.

U Spektrumu se ne edresema koje poziveju restart naredbe nalaze bezične ROM rutine čime se olakšava njihovo pozivanje.

Upotreba restert naredbi je ilustrovane primerom 9.

Naredbe povratke (Return)

Naredbe ove grupe se primenjuju za vraćanje iz mašinskih potprogreme.

Izvršavanjem neredbe povratka, sa steka se uzima adresa lokacije se koje je usledio skok u potprogrem. Ta adrese se stevlja u programski brojač, čime se nastavlje izvršavanje programa.

Naredba povratke **RET** sa **CALL** naredbom čini par za odlazak i povratak iz mašinskih potprograma ili rutine prekida. Takode, **RET** naredbom se izvršava povrtak iz mešinskog u bejzik program.

Iz tabele 4–9 vidi se da **RET** naredba zauzima jedan bajt memorije i de se njeno izvrševanje može uslovıti istim uslovime kao i za **JP** i **CALL** naredbe.

Mnemonička oznaka naredbe povratka uz uslov da je rezultet prethodne operecije jednek nuli, to jest da je Z bit jednak 1 je

RET Z

Ostale mnemoničke oznake su anelogne prethodno izloženom

Postoje još dve posabne naradbe povratka. To su povratak iz prakida **(RETI)** i povratak iz nemaskirajućeg prakida **(RETN)**.

Povratak iz prekida se primenjuje na kraju programa za servisiranje prekida, a povratak iz nemaskirajućeg prakida na kraju programa za servisiranje nemaskirajućeg prekida. Obe naradbe se od strana procasora tretiraju kao povratak **(RET)**. Razlika je u tome što **RETI** omogućava da periferna jadinice prepoznaju ovu naredbu.

4. NAREDBE PREBACIVANJA BLOKOVA I PRETRAŽIVANJA

Naredbe ove grupe omogućuju prebacivanje sadržaja jednog bloka memorije u drugi, a takođe i pretraživanje blokova mamorije u cilju nalaženja određenog sadržaja. Ovo su vaoma praktične i često primenjivane naredbe.

Naradbe prebacivanja korista tri ragistar parа na sledaći način:

HL ragistar par ukazuje na mamorijsku lokaciju sa koje se uzima podatak

DE registar par ukazuje na memorijsku lokaciju na koju se prabacuje podatak

BC registar par je brojač bajta

Izvršavanje naredba **LDIR** (load, increment and repeat), kojom se prebacuje blok memorije, obavlja se na sledeći način. Sadržaj memorijska lokacije na koju ukazuje HL registar par prabacuje se u memorijsku lokaciju određenu DE registar parom. Zatim sa vrednost i sadržaja HL i DE registar parova povećavaju za jadan dok se sadržaj BC registar para smanjuje za jedan. Cao postupak sa ponavlja sve dok brojač bajta, BC registar par, na dođa do nule. Tima je jednom mašinakom naradbom izvršeno prabacivanja blokova memorije čija je dužina bila sadržana u BC registar paru.

U slučaju kada je potrabno je da sa izvrši naka druga oparacija između prebacivanja pojedinih bajta, primanjuje sa naradba **LDI** (load and increment). Razlika između ove i **LDIR** naredba ja u tome što se posle prenosa jednog bajta ciklus prenosa neće ponoviti.

Naredbe **LDDR** i **LDD** su veoma slične sa **LDIR** i **LDI** naredbama. Jadina je razlika u tome da se ragistar parovima **HL** i DE smanjuje vrednost sadržaja za jadan posle svakog prebacivanja. Time prebacivanje bloka počinja na višoj adresi, a završava se na nižoj.

Mnemonika i kodovi naredbi su dati u tabeli 4–11, a program koji ilustruje ovu grupu naredbi je dat primerom 11.

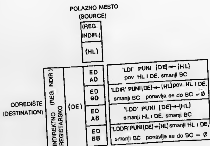


Tabela 4–11 Naredbe prebacivanja blokova

Postoje četiri naredbe pretraživanja bloka memorije pomoću kojih se obavlje nalaženje željenog sedržaja. **CPIR** (compare, increment and repeat) poradi sadržaj akumulatora sa sadržajem memorijske lokacije na koju ukazuje HL registar per. Rezultat operecije se postavlja u odgovarajući bit stanja, sadržaj HL registar para povećava vrednost za jedan, dok se vrednost sedržaja BC registar pera smanjuja ze jadan. Ceo postupek se ponavlja sve dok se u BC registar paru ne pojavi nule, ili dok se ne pronade vrednost sadržaja neke memorijske lokacije koja je jednaka sadržeju A registra. Na ovaj način se moža vršiti i pretraživanja cele memorija da bi sa pronašao jedan osmobitni podatak.

Naredba **CPI** (compare and increment) sa razlikuje od **CPIR** samo po tome što sa ne obavlja automatsko ponavljanje do ispunjavanja uslove. Izvršava se poredenje, postavljanje registra stanja i povećanje, odnosno smanjenja vrednosti registar parova HL i BC. Ponevljanje celog postupka zahteva ponovno pozivenja **CPI** nerodbe. Ova se naredba zbog toga koristi kad je potrebno izmedu oparacija poredenja izvršiti neku drugu operaciju.

Naredbe **CPDR** (compare, decrement and repeat) i **CPD** (compare end decrement) su slične naredbe. Jedina je razlika što

se u HL registar paru vrednost smanjuje za jedan pri svakoj operaciji poređenja. Na taj način pretraživanje memorije počinja od najviše adrese bloka.

Mnemonika i kodovi naredbi pratraživanja su dati u tabeli 4–12, a odgovarajući primar je 12.

MESTO PRETRAŽIVANJA (SEARCH LOCATION) (REG INDIR) (HL)	
ED A1	'CPI' pov HL, smanji BC
ED B1	'CPIR' pov HL, smanji BC ponavlja se do BC = Ø ili nalaženja
ED A9	'CPD' smanji HL i BC
ED B9	'CPDR' smanji HL i BC ponavlja se do BC = Ø ili nalaženja

Tabela 4–12 Naredbe pratraživanja blokova

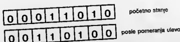
5. NAREDBE ROTACIJE I POMERANJA

Jedna od glavnih osobina procasora Z80 je mogućnost rotacija i pomeranja bita u akumulatoru, u registrima opšta namene i u mamorijskim lokacijama. To može biti od značaja kada se uzme u obzir da se pomeranjem bita za jedno mesto ulevo vrši množenje sa dva, a pomaranjam udasno postiže se deljenja sa dva.

Primar množanja sa dva vrednosti sadržana u jednoj memorijskoj lokaciji prikazan je na slici 4–3.

Pra pomeranja, vrednost sadržana u jednoj lokaciji iznosila je 26, a posla pomeranja u levo za jadan bit, vrednost je 52.

Naredba rotacije takođe utiču na pokazatelj prenosa, C bit. Ta značajna osobina omogućava njihovu primenu i u logičkim operacijama.



Slika 4–3 Simbolički prikaz operacije pomeranja ulevo

Sve naredbe rotacije su prikazane u tabeli 4–13. Na levoj strani je dat komentar onoga što se dogodа sa nultim bitom, sedmim bitom i C-bitom. Ostali biti se pomeraju za jedno mesto u levom ili u desno. Sa desne strane su date odgovarajuće mnemonička oznake i nazivi naradbi.

Tabela pored odgovarajućih kodova prikazuje i nad kojim registrima i memorijskim lokacijama je moguće izvršiti određene rotacije. Uočava se da su to akumulator, svaki registar opšte namene ili bilo koja memorijska lokacija adresirana sadržajem HL, IX ili IY registar parovima.

Posebno treba izdvojiti četiri jednobajtne naredbe rotacije A registra. Njihove mnemoničke oznake su **RLCA, RRCA, RLA** i **RRA**. Postoje i dve specijalizovane naredbe za ubrzavanje BCD aritmetike. To su naredba pomoću kojih se vrši rotiranje sadržaja akumulatora i dveju memorijskih lokacija ukazanih HL registar parom. Šematski prikaz svih tipova rotacije prikazan je u sklopu tabele 4–13.

Sadržaj C bita F registra zavisi od rezultata operacije u slučaju svih naredbi sem za naredbe sa ozankom **RRD** i **RLD**. Pokazatelji nule, parnosti, premašenja i znaka dobijaju vrednosti koje zavise od rezultata operacije. Ova konstatacija ne važi za jednobajtne naredbe koje se odnose na A registar.

Mogućnosti primene naredbi rotacije su velike, a neke od njih su prikazane u primerima 9 i 15.

6. NAREDBE MANIPULACIJE BITIMA

Naredbe ove grupa omogućavaju da procesor Z80 izvrši postavljanje na nulu, postavljanje na jedinicu ili proveru vrednosti svakog bita posebno, kako u svim registrime, tako i u svim memorijskim lokacijama.

240 naredbi ovog tipa je prikazano u tabeli 4–14 gde je izvršena podela na sledeće tri grupe:

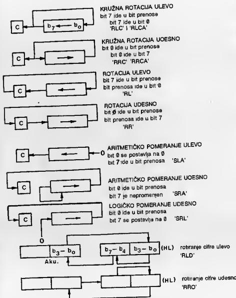
a) Naredbe postavljanja bita na nulu (reset)

POLAZNA MESTA I ODREDIŠTA

TIP ROTACIJE ILI POMERANJA	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
'RLC'	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 08	DD CB d D6	FD CB d 06
'RRC'	CB 0F	CB 08	CB 09	CB 0A	CB 08	CB 0C	CB 0D	CB 0E	DD CB d DE	FD CB d 0E
'RL'	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
'RR'	CB 1F	CB 16	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
'SLA'	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
'SRA'	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
'SRL'	CB 3F	CB 36	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
'RLD'								ED BF		
'RRD'								ED 67		

	A
RLCA	07
RRCA	0F
RLA	17
RRA	1F

Tabela 4–13 Naredbe rotacije i pomeranja



Slika 4–4 Simbolički prikaz operacija rotacija i pomeranja

	BIT	REGISTARSKO ADRESIRANJE (REGISTER ADDRESSING) A	B	C	D	E	H	L	(REG INDIR) (HL)	INDEKSIRANO (INDEXED) (IX+d)	(IY+d)
POSTAVLJANJE NA NULU (RESET BIT RES)	0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
POSTAVLJANJE NA JEDINICU (SET BIT SET)	0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE
PROVERA BITA (TEST BIT)	0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
	2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
	4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
	6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E

Tabela 4-14 Naredbe manipulacije bitima

Ove naredbe omogućuju da se odredeni bit akumuletora, registra opšte namene ili memorijskih lokacije edresiranih HL, IX i IY registrima postave na a nulu. Ako je prethodno stanje već bilo Ø, izvršavanjem naredbe ništa se neće promaniti.

Mnemonika naredbe za postavljanje prvog bita akumulatora na nulu je date primerom

RES 1,A

Kôd ove naredbe je C88Fh što se vidi iz tabele.

b) Naredbe postavljanja bite ne jedinicu (set)

Ove naredbe omogućavaju da se jedan bit, određen kao u prethodnom slučaju, postavi na jedinicu. Ukoliko je već bio na jedinici, izvršavanjem naredbe ništa se neće promeniti.

Četvrti bit memorijske lokacije adresirane HL registar parom postavlja se na jedinicu naredbom

SET 4,(HL)

c) Naredbe provere bite (test)

Neredbama provere se vrši ispitivenje vrednosti željenog bita akumuletora, registra opšte namene ili memorijskih lokacija adresiranih HL, IX ili IY registar parovima.

Ako je vrednost bite jednaka nuli, pokazatelj nule (Z) se postavija na jedinicu i obrnuto. Upotreba naredbi provere omogućeva da se pojedinečni biti upotrebe kao biti stanje. Na taj način se ove naredbe uobičajeno koriste za obavljanje grananja u programu.

Mnemonika naredbe provere je data na primeru provere sedmog bita memorijske lokacije adresirane IX registar parom.

BIT 7,(IX)

Primene naredbi iz ove grupe se mogu videti u primeru 10.

7. ULAZNO–IZLAZNE NAREDBE

Naredbe ove grupe (engl. input-output) omogućavaju razmenu podetake izmedu mikroprocesora i ulazno-izlazne jedinice. Adresiranje periferne jedinice, koje može da šelje ili da prima podetke, može bit eposolutno ili registarsko indirektno.

Pri apsolutnom adresiranju neredbame **IN A,(N)** i **OUT (N), A** izvršava se razmena podetaka izmedu akumulatora i periferne jedinice. Na nižih 8 bite adresne megistrele (A_0–A_7) postrevlje se vrednost konstante N, dok se ne viših 8 bita (A_8–A_{15}) postavija vrednost iz akumulatora.

U registarskom indirektonom adresiranju svaki interni registar proceeora može razmenjivati podatke sa perifemom jedinicom. Ad-

ODREDIŠTE (DESTINATION)			ADRESA ULAZNE JEDINICE: NEPOSREDNO (IMME) (n)	ADRESA ULAZNE JEDINICE: (REG INDIR.) (c)	
ULAZ (INPUT) 'IN'	REGISTARSKO (REGISTER ADDRESSING)	A	DB n	ED 78	
		B		ED 40	
		C		ED 48	
		D		ED 50	
		E		ED 58	
		H		ED 80	
		L		ED 68	
'INI' ULAZ pov. HL, smanj. B	INDIREKTNO (REG. INDIR.) REGISTARSKO	(HL)		ED A2	NAREDBE ULASKA BLOKA
'INIR' ULAZ pov. HL, smanj. B ponavlja se ako B ≠ O				ED B2	
'IND' ULAZ smanj. HL, smanj. B				ED AA	
'INDR' ULAZ smanj. HL, smanj. B ponavlja se ako B ≠ O				ED BA	

Tabela 4–15 Ulazne naredbe

POLAZNO MESTO (SOURCE)

			REGISTAR (REGISTER)							(REG INDIR.)
			A	B	C	D	E	H	L	(HL)
IZLAZ 'OUT'	NEPOSREDNO (IMME.)	(n)	D3 n							
	(REG INDIR.)	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69	
'OUTI' – IZLAZ pov. HL, smanj. B	(REG. INDIR.)	(C)								ED A3
'OTIR' – IZLAZ pov. HL, smanj. B ponavlja se ako B ≠O	(REG INDIR.)	(C)								ED B3
'OUTD' – IZLAZ smanj. HL, smanj. B	(REG INDIR.)	(C)								ED AB
'DTDR' – IZLAZ smanj. HL, smanj. B ponavlja se ako B ≠O	(REG INDIR.)	(C)								ED BB

ADRESA IZLAZNE JEDINICE

NAREDBE IZLAZA BLOKA

Tabela 4–16 Izlazne naredbe

resiranje se obevlje tako što se sadržaj C registra postavlja ne nižih 8 bita, e sadržej B registra u viših 8 bita edresne megistrele. Na taj nečin se može adresirati 65536 spoljašnjih iokacija. Moguće je razmena 8-bitnog podatka sa svakim registrom opšte namene.

Ulazno-izlazne naredbe prebacivanja blokova, koje se nelaze u grupi ulazno-izlaznih naredbi, slične su sa naredbama prebecivanja blokova. Razlika je u tome što se HL registar par koristi za adresiranje lokacije sa koje se primaju podaci (u izlaznim neredbama) ili lokacije na koju se prebacuju podaci (u ulaznim naredbeme). Pri tome se B registar koristi kao brojač bajta što omogućuje prenos do 256 bajta. Adresiranje se obevlje preko sadržaje registera C i B pri čemu je u C registru smešten niži bejt, a u B registru viši bajt adresne megistrale.

Tebele 4–15 i 4–16 prikazuju moguće ulazno-izlazne naredbe sa odgoverajućim kodovima. Primeri 13 i 14 ilustruju primenu naredbi ove grupe.

8. NAREDBE KONTROLE PROCESORA

Poslednja tebela naredbi, tabela 4–17, prikazuje sedam kontrolnih naredbi opšte nemene.

NOP (no operation) naredba ne izvrševa ništa. Zauzime jeden bajt memorije i dovodi do kratkotrajne pauze u programu koji sadrži ovu naredbu.

Sledećih šest neredbi se odnose ne prekide (eng. interrupt). Do prekida dolazi pri pojavi odgovarejućeg signela ne izvodu mikroprocesora koji je za to predviđen (INT ili NMI). Tada se obustavlja izvrševanje tekućeg programe i prelazi na izvršavanje programa koji je u tu svrhu napisan (servisna rutina prekida). Posle njegovog izvršavanja vraća se na izvršavanje prekinutog programa. Postoje dve vrste prekida: onaj koji se može onemogućiti, maskirati (INT) i onaj koji se ne može onemogućiti tj. meskireti (NMI – non maskable interrupt) odgovarajućom naredbom.

HALT (hait) neredbe onemogućuje procesor u daljem izvršavenju naredbi sve do naiiaske signela prekida INT ili NMI.

DI (disable interrupt) naredba onemogućava prekid redovnog programa procesora. To znači da neće doći do prekide kada periferna jedinice uputi signal maskirajućeg predide INT (maskable interrupt).

'NOP'	00
'HALT'	76
'(DI)' ONEMOGUĆENJE PREKIDA	F3
'(EI)' OMOGUĆAVANJE PREKIDA	F8
'IM0' POSTAVLJANJE NAČINA PREKIDA 0	E0 46
'IM1'	ED 56
'IM2'	ED 5E

Tabela 4–17 Naredbe kontrole procesora

EI (enable interrupt) naredba omogućava procasoru prihvatanje INT signala. Ova naredba, zajedno sa pradhodnom, omogućava selektivan izbor parioda u toku izvršavanja programa u kojima na treba dozvoliti prekid.

Z80 može da odgovori na zahtev za prekidom na tri različita načina: način Ø, način 1 i način 2 (engl. mode Ø, mode 1, mode 2). Način rada se bira naredbama **IMØ**, **IM1** i **IM2**).

IMØ naredba prevodi procesor Z80 u nulti način rada. Tada periferna jedinica može postavljati kodove naredbi na magistralu podataka, a procesor će ih izvršavati.

IM1 naredba prevodi procesor u prvi način rada. Pojavom signala prekida procesor će izvršiti restart na lokaciji ØØ38h. Ovo je način rada Spektruma. ULA kolo predstavlja periferrnu jedinicu za procesor Z80. ULA 50 puta u sekundi šalje signal prekida u procesor. Svaki put kada primi signal prekida, procesor će izvršiti restart na lokaciju ØØ38h odakle počinju naredbe koje omogućavaju izvršavanja rutine za očitavanje tastature. Ta rutina povećava sadržaj sistamske promenljive FRAMS i ispituje da li je neki tastar pritisnut. Kôd pritisnutog tastara se smešta u sistemsku promenljivu LAST K i postavlja se pati bit promenljive FLAGS (23611) za indikaciju novop-

ritisnutog tastera. Završivši rutinu, procesor se vraća na mesto prekida i nastavlja sa izvršavanjem programa.

IM2 naredba prevodi procesor u drugi način rada. Ovaj način rada pruža najveće mogućnosti. Posle dobijanja signala prekida, procesor očitava 16-bitnu vrednost. Gornji bajt adrese ta vrednosti je sadržaj registra I, a donji bajt adrese se čita sa periferne jedinice



(taj podatak je ranije upisan u nju). Kod ULA taj podatak je FFh.

Kada dođe zahtev za nemaskirajućim prekidom, mikroprocesor će izvršiti skok na memorijsku lokaciju Ø066h. Ova vrsta prekida ne može se u Spektrumu iskoristiti zbog greške u ROM-u na adresi Ø06Eh, usled čega pri nemaskirajućem prekidu dolazi do reinicijalizacije sistema.

U trenutku pojave signala prekida, programski brojač PC stavlja svoj sadržaj na stek i preuzima nov sadržaj sa adresirane lokacije RAM-a. Po završetku programe predviđenog za slučaj prekida, sa steka se vraća sadržaj PC registra obezbeđujući nastavljanje prekinutog programa.

U radu sa Spektrumom pri upotrebi prekida posebnu pažnju treba posvetiti ponovnom dodeljivanju vrednosti upotrebljenim registrima i promenljivama.

Primeri 5 i 15 prikazuju upotrebu naredbi ove grupe

4–6 POKAZATELJI STANJA

U svakom od dva F (flag) registra procesora Z80 koriste se šest bita za označavanje različitih stanja procesora. Četiri od njih se mogu proveravati, testirati i upotrebljavati kao uslov za grananje u programu.

Biti koji se mogu proveravati su:

1. Pokazatelj prenose C (carry flag). Sabiranjem dva osmobitna broja rezultat može izaći iz opsega osmobitnih brojeva. Tada se javlja potreba za prenosom u dodatni bit i C bit će biti postavljen na jedinicu. Isto će se dogoditi i u slučaju oduzimanja, ako rezultat izađe iz

opsega osmobitnih brojeva. Ovaj bit sa postavlja na jedinicu i pri izvršavanju naredbi rotacija i pomeranja.

2 Pokazatelj nula Z (zero flag). Ovaj pokazatelj se postavlja na jedinicu samo ako je rezultat operacija koji sa stavlja u akumulator jednak nuli. U protivnom Z bit se postavlja na nulu.

3 Pokazatelj zneka S (sign. flag). Pokazateljem znaka ukazuje se da je rezultat izvršenih oparacija negativan. U formi komplementa dvojke krajnji levi bit (bit 7 u osmobitnom broju) označave znak broja. Bit 5 F registra će prema tome imati vrednost najtežeg bita akumulatora.

4 Pokazatelj parnosti i premašenja P/V (parity/overflow). Ovaj pokazatelj ima dvostruku namenu. Označava da li je za razultat operacije, koji ja u akumulatoru, zadovoljena parnost i da li je došlo do premašenja u slučaju aritmetičkih operacija u komplementu dvojke. Ako je rezultat operacije koji je predstavljen u komplementu dvojke van opsega brojeve od −128 do +127 bit premašanja sa postavlja na jedinicu. U sledečem primeru ja rezultat sabiranja tačan u apsolutnoj binarnoj formi, ali ne i u formi komplemanta dvojka. Bit premašenja se tada postavlja na jedinicu.

```
  50    00110010
+ 99    01100011
 149    10010101 = −21 u komplemantu dvojka
```

U logičkim operacijama P/V bit se postavlja na jedinicu ako 8-bitni binarni broj ima paran broj bita sa vrednošču jedan. Za slučaj neparnog broja jedinica u akumulatoru, ovaj bit ostaje nula. Za sadržaj akumulatora 10101100 bit parnosti se postavlja na jedinicu, a za 11110111, na nulu.

Sledača dve bita se ne mogu provaravati i oba se korista u BCD aritmatici.

1. Pokazatelj poluprenosa (H). Ovo je u stvari pokazatelj prenosa u radu sa decimalnim ciframa predstavljenih pomoću četiri bita. Dobija se prenosom ili pozajmicom iz četiri bita manja težine. Upotreba DAA naredbe omogućava iskoriščavanje ovog bita za korekciju razultata BCD sabiranja ili oduzimanja.

2 Pokazatelj oduzimanja (N). S obzirom da je postupak za korekciju BCD rezultata različit za sabiranje ili oduzimanje, ovaj pokazatelj se koristi da ukaže koja vrsta naredbe je izvršena poslednja (sabiranja ili oduzimanje).

Raspored bite stanje u F registru je

D7	D6	D5	D4	D3	D2	D1	D0
S	Z		H		P/V	N	C

Prazna polje ukazuju da ti biti nisu upotrebljeni. Konačno tabela 4–18 prikazuje kako na svaki pokazatelj stanja utiču naredbe procesora.

Naredbe	Biti pokazatelji						Komentar
	C	Z	P/V	S	N	H	
ADD A,s; ADC A,s	+	+	V	+	0	+	8-bitno sabiranje i sabiranje sa prenosom
SUB s; SBC A,s; CP s NEG	+	+	V	+	1	+	8-bitno oduzimanje, oduzimanje sa prenosom, poređenje i negacija akumulatora
AND s	0	+	P	+	0	1	Logičke operacije
OR s; XOR s	0	+	P	+	0	0	
INC s		+	V	+	0	+	8-bitno povečanje
DEC s		+	V	+	1	+	8-bitno smanjenje
ADD DD,ss	+				0	X	16-bitno sabiranje
ADC HL,ss	+	+	V	+	0	X	16-bitno sabiranje sa prenosom
SBC HL,ss	+	+	V	+	1	X	16-bitno oduzimanje sa prenosom
RLA; RLCA; RRA; RRCA	+				0	0	Rotacija akumulatora
RL s; RLC s; RR s; RRC s; SLA s; SRA s; SRL s	+	+	P	+	0	0	Rotacija i pomeranje lokacije s
RLD; RRD		+	P	+	0	0	Rotecija cifara levo i desno
DAA	+	+	P	+		+	Decimalno podešavanje akumulatora
CPL					1	1	Komplementiranje ekumulatora

SCF	1				0	0	Postavljanje prenosa na jedinicu
CCF	+				0	X	Komplementiranje bite prenosa
IN r,(C)		+	P	+	0	0	Ulazna naredbe
INI; IND; DUTI; OUTD		+	X	X	1	X	Ulazno-izlazna nar. blokova
INIR, INDR; DTIR; DTDR		1	X	X	1	X	Z=0 ako je B≠0 u protivnom P/V=0
LDI; LDD		X	+	X	0	0	Prebacivanja blokova
LDIR, LDDR		X	0	X	0	0	P/V=1 ako je BC≠ u protivnom P/V≠0
CPI; CPIR; CPD; CPDR		+	+	+	1	X	Pretraživanje blokova Z=1 ako je A=(HL), u protivnom Z=0. P/V=1 ako je BC≠0, u protivnom P/V=0
LD A,I; LD A,R		+	IFF	+	0	0	Vrednost flip flopa za dozvolu prekida (IFF) se prebacuje u P/V bit
BIT b,s		+	X	X	0	1	Z bit sa postavlja na vrednost bita b lokaci-je s
NEG	+	+	V	+	1	+	Negecije akumulatora

Tabela 4-18 Pokazatelji

U tabeli su upotrebljene sledeće oznake:

- r   Bilo koji registar procesora A,B,C,D,E,H,L
- s   Bilo koja 8-bitna lokacija ili vrednost za načine adresiranja predviđene naredbom
- ss   Bilo koja 16-bitna lokacija ili vrednost za načine adresiranja predviđene naredbom
- \+   Pokazatelj se postavlja zavisno od razulteta operacija
- 0   Pokazatelj se postavlje na nulu
- 1   Pokazatelj se postevlja na jedinicu

X Pokazatelj zauzima neodređeno stanje
prazno polje Stanje pokazatelja se ne menja

4-7 SPISAK NAREDBI PROCESORA Z80

Naredna tabela donosi spisak mnemoničkih oznaka svih naredbi procesora Z80. Takođe je za svaku naredbu dat broj bajta koji zauzime u memoriji (kolona BB), broj mašinskih ciklusi i T stanja potrebnih za izvršenje (kolone BC i T). Mašinski ciklusi i T stanja su obrađeni u 6. delu knjige u okviru opisa vremenski dijagrama.

Mnemonika	Naredba	BB	BC	T
ADC HL, ss	Sabiranje sa prenosom reg.para ss sa HL	2	4	15
ADC A,s	Sabiranje sa prenosom operanda s i akumulatora			
ADD A,n	Sabiranje akumuletora sa n	2	2	7
ADD A,r	Sabiranje registra r se akumuletorom	1	1	4
ADD A,(HL)	Sabiranje lokacije (HL) se akumulatorom	1	2	7
ADD A,(IX + d)	Sabiranje lokacije (IX + d) se akumulatorom	3	5	19
ADD A,(IY + d)	Sabiranje lokacije (IY + d) sa akumulatorom	3	5	19
ADD HL,ss	Sebiranje reg.para ss se HL	1	3	11
ADD IX,pp	Sabiranje reg.para pp se IX	2	4	15
ADD IY,rr	Sabiranje reg.para rr se IY	2	4	15
AND s	Logičko množenje između s i akumul.			
BIT b,(HL)	Provera bita b na lokaciji (HL)	2	3	12
BIT b,(IX + d)	Provera bita b na lokaciji (IX+d)	4	5	20
BIT b,(IY + d)	Provera bita b ne lokaciji (IY+d)	4	5	20
BIT b,r	Provere bita b registra r	2	3	8
CALL cc,nn	Odlazak ne potprogram na adresi nn uz uslov cc	3	3	10

	(eko je cc tačno)	3	5	17
CALL nn	Bezuslovni odlazak na potprogram na adresi nn	3	5	17
CCF	Komplementiranje bite prenosa (C)	1	1	4
CP s	Poređenje akumulatora sa operendom s			
CPD	Poređenje lokecije (HL) i akum.smanjenje HL i BC	2	4	16
CPDR	Poređenje lokacije (HL) i akum.smanjenje HL i BC ponavljanje do BC=∅	2	5	21
	eko je BC=∅ ili A=(HL)	2	4	16
CPI	Poređenje lokacije (HL) u akum. povećanje HL i smanjenje BC	2	4	16
CPIR	Poredenje lokecije (HL) i akum. povećenje HL, smenjenje BC, ponavljanje do BC=∅	2	5	21
	eko je BC=∅ ili A=(HL)	2	4	16
CPL	Komplementiranje sadržaja akumuletora	1	1	4
DAA	Prevođenje vrednosti akum. u BCD oblik	1	1	4
DEC m	Smanjenje operanda m			
DEC IX	Smenjenje IX	2	2	10
DEC IY	Smanjenje IY	2	2	10
DEC ss	Smanjenje registar para ss	1	1	6
DI	Onemogućenje prekida	1	1	4
DJNZ e	Smanjenje B i relativni skok	2	2	8
	ako je B=∅			
	ako je B≠∅	2	3	13
EI	Omogućenje prekide	1	1	4
EX (SP), HL	Zamena lokacija (SP) i HL	1	5	19
EX (SP), IX	Zemena lokacija (SP) i IX	2	6	23
EX (SP), IY	Zamena lokacija (SP) i IY	2	6	23
EX AF,AF'	Zamene sadržaja AF i AF'	1	1	4
EX DE,HL	Zamena sadržaje DE i HL	1	1	4

EXX	Zamena sadržaja BC, DE, HL sa sadržajem BC', DE', HL'	1	1	4
HALT	Čekanje prekida ili resate	1	1	4
IMO	Način rada Ø ne pojevu prekide	2	2	8
IM 1	Način rada 1 na pojavu prekida	2	2	8
IM 2	Način rada 2 na pojavu prekida	2	2	8
IN A,(n)	Punjenje akum. sa spoljna jadinice n	2	3	11
IN r,(c)	Punjenje registra r sa spoljne jed. (C)	2	3	12
INC (HL)	Povečanje lokacija (HL)	1	3	11
INC IX	Povaćanje IX	2	2	10
INC (IX+d)	Povećanje lokacije (IX+d)	3	6	23
INC IY	Povečanje IY	2	2	10
INC (IY+d)	Povećanje lokacija (IY+d)	3	6	23
INC r	Povećanja registra r	1	1	4
INC ss	Povećanje registar para ss	1	1	8
IND	Punjenje lokacije (HL) sa spoljne jedinica (C), smanjenje HL i B	2	4	18
INDR	Punjenje lokacije (HL) sa spoljne jedinica (C), smanjenje HL i smanjenje B, ponavljanja do B=Ø	2	5	21
	eko je B=Ø	2	4	16
INI	Punjenja lokecije (HL) se spoljne jed, (C) povečanja HL i smanjanje B	2	4	16
INIR	Punjenja lokacije (HL) sa spoljne jedinice (C) povačanje HL i smanjenje B, ponavljanje do B=Ø	2	5	21
	ako ja B=Ø	2	4	16
JP (HL)	Bezuslovni skok na (HL)	1	1	4
JP (IX)	Bezuslovni skok na (IX)	2	2	8
JP (IY)	Bazuslovni skok na (IY)	2	2	8
JP cc,nn	Skok na lokaciju nn uz uslov cc	3	3	10
JP nn	Bezuslovni skok na lokeciju nn	3	3	10

JR C, a	Relativni skok na PC + e ako je prenos 1	2	2	7
	ako je C = 1	3	1	12
JR e	Bezuslovni ralativni skok na PC + e	2	3	12
JR NC, a	Relativni skok na PC + a ako je prenos 0	2	3	12
	eko je C = 1	2	2	7
JR NZ, e	Ralativni skok na PC + e pokazatelj nule različit od 0 (Z=0)	2	3	12
	ako je Z = 1	2	2	7
JR Z, e	Ralativni skok na PC + e ako je pokazataij nule jednak 0 (Z=1)	2	3	12
	ako je Z = 0	2	2	7
LD A, (BC)	Punjenje akum. sa lokacijom (BC)	1	2	7
LD A, (DE)	Punjenje akum. sa lokacijom (DE)	1	2	7
LD A, I	Punjenje akumulatora sa I	2	2	9
LD A, (nn)	Punjenje akumulatora sa lok. (nn)	3	4	13
LD A, R	Punjenje akumulatora sa registrom R	2	2	9
LD (BC), A	Punjenje lokacija (BC) sa ekumul.	1	2	7
LD (DE), A	Punjenje lokacija (DE) sa akumul.	1	2	7
LD (HL), n	Punjenje lokacija (HL) sa vrednošću n	2	3	10
LD dd, nn	Punjenje registar para dd sa vrednošću nn	3	3	10
LD HL, (NN)	Punjenje HL sa lokacijom (nn)	3	5	16
LD (HL), r	Punjenje lokacije (HL) sa registrom r	1	2	7
LD I, A	Punjenje I sa ekumulatorom	2	2	9
LD IX, (nn)	Punjenje IX sa lokacijom nn	4	6	20
LD (IX + d), n	Punjenje lok. (IX + d) sa vradnošću n	4	5	19
LD (IX + d), r	Punjenje lok. (IX + d) sa vrednošću registra r	3	5	19
LD IY, nn	Punjenje IY sa vrednošću nn	4	4	14
LD IY, (nn)	Punjenje IY sa lok. (nn)	4	6	20
LD (IY + d), n	Punjenje lokacija (IY + d) sa vradnošću n	4	5	19
LD (IY + d), r	Punjenje lokacije (IY + d) sa vradnošću registra r	3	5	19
LD (nn), A	Punjenje lokacija (nn) sa akumulatorom	3	4	13
LD (nn), dd	Punjenje lokacija (nn) sa registar parom dd	4	6	20

LD (nn), HL	Punjenje lokacije (nn) sa HL	3	5	16
LD (nn), IX	Punjenje lokacije (nn) sa IX	4	6	20
LD (nn), IY	Punjenje lokacije (nn) se IY	4	6	20
LD R, A	Punjenje R se akumulatorom	2	2	9
LD r, (HL)	Punjenje registra r sa lokacijom (HL)	1	2	7
LD r, (IX + d)	Punjenje registre r sa lokacijom (IX + d)	3	5	19
LD r, (IY + d)	Punjenje registre r sa lokacijom (IY + d)	3	5	19
LD r, n	Punjenje registra r sa vrednošću n	2	2	7
LD r, r'	Punjenje registra r sa registrom r'	1	1	4
LD SP, HL	Punjenje SP sa HL	1	1	6
LD SP, IX	Punjenje SP sa IX	2	2	10
LD SP, IY	Punjenje SP sa IY	2	2	10
LDD	Punjenje lokacije (DE) sa lok. (HL), smenjenje DE, HL i BC	2	4	16
LDDR	Punjenje lokacije (DE) sa lok. (HL), smenjenje DE, HL i BC, ponavljenje do BC = 0	2	5	21
	ako je BC = 0	2	4	16
LDI	Punjenje lokacije (DE) sa lok. (HL), povećanje DE, HL, smanjenje BC	2	4	16
LDIR	Punjenje lokacije (DE) sa lok. (HL), povećanje DE, HL, smenjenje BC, ponavljanje do B = 0	2	5	21
	ako je BC = 0	2	4	16
NEG	Komplement dvojke ekumulatora	2	2	8
NOP	Logičko ILI između akum. i operenda s	1	1	4
OR s				
OTDR	Punjenje izlazne jedinice C sa lok. (HL), smanjenje HL i B, ponavljanje do B = 0	2	5	21
OTIR	Punjenje izlazne jedinice C sa lok. HL, povećenje HL, smanjenje B, ponavljanje do B = 0	2	5	21
	ako je B = 0	2	4	16
OUT (C), r	Punjenje izlazne jedinice C sa registrom r	2	3	12
OUT (n), A	Punjenje izlazne jedinice (N) sa akumulatorom	2	3	11

OUTD	Punjenje izlazna jedinice (C) sa lokacijom (HL) povaćanje HL i smanjenje B	2	4	16
OUTI	Punjenje izlazne jad. (C) sa lok. (HL), povećanje HL i smanjenje B	2	4	16
POP IX	Punjanje IX sa poslednjom vradnošću sa steka	2	4	14
POP IY	Punjenje IY sa poslednjom vrednošću sa steka	2	4	14
POP qq	Punjanje reg.para qq sa poslednjom vradnošću sa steka	1	3	10
PUSH IX	Punjenje staka sa IX	2	4	15
PUSH IY	Punjenje staka sa IY	2	4	15
PUSH qq	Punjenje steka sa registar parom qq	1	3	11
RES b,m	Postavljanje na nulu bita b operanda m			
RET	Povratak sa potprograma	1	3	10
RET cc	Povratak sa potprograma uz uslov cc	1	1	5
	(ako je cc tačno)	1	3	11
RETI	Povratak iz prekida	2	4	14
RETN	Povrat.iz predika koji se ne može maskirati	2	4	14
RL m	Rotacija lavo preko bita prenosa operanda m			
RLA	Rotacija levo akum. preko bita prenosa	1	1	4
RLC (HL)	Rotacija lokacije (HL) levo kružno	2	4	15
RLC (IX+d)	Rotacija lokacije (IX+d) lavo kružno	4	6	23
RLC(IY+d)	Rotacija lokacije (IY+d) levo kružno	4	6	23
RLC r	Rotacija registra r levo kružno	2	2	8
RLCA	Rotacija cifara lavo kružno akum.	1	1	4
RLD	Rotacija cifara levo i desno izmedu akum. i lokacije (HL)	2	5	18
RR m	Rotacija desno preko bita, prenosa operanda m			

RRA	Rotacija desno akum. preko bita prenosa	1	1	4
RRC m	Rotecija operanda m desno kružno			
RRCA	Rotacija desno kružno akum.	1	1	4
RRD	Rotacija cifara desno i levo između akumulatora i lokacije (HL)	2	5	18
RST p	Restart ne lokaciji p	1	3	5
SBC A,s	Oduzimanje operanda s od akum. sa prenosom			
SBC HL, ss	Oduzimanje registar para ss od HL sa prenosom	2	4	15
SCF	Postavljanje bita prenosa na jedinicl	1	1	4
SET b,(HL)	Postavljanja bita b lokacije (HL)	2	4	15
SET b,(IX+d)	Postevij. bita b lokacije (IX+d)	4	6	23
SET b,(IY+d)	Postavij. bita b lokacije (IY+d)	4	6	23
SET b,r	Postavij. bita b registra r	2	2	8
SLA m	Pomeranje operanda m levo antmetički			
SRA m	Pomeranje operanda m desno aritmetički			
SRL m	Pomeranje operanda m desno logički			
SUB s	Oduzimanje operanda s od ekum.			
XOR s	Isključivo ILI između akum. i operenda s			

4–8 PRIMERI PROGRAMIRANJA NA MAŠINSKOM JEZIKU

U pretnodnim poglavljima su izložene neophodne postavke za razumevanje mašinskih programa. U ovom poglavlju će se prikazati nekoliko mešinskih programe u cilju ilustrovanja korišćenja mašinskih naredbi keo i nećine formiranje i korišćenja progrema na mešinskom jeziku.

Mašinski progremi se mogu unositi u računer na više načina. Jeden od njih je pogodan za unošenje primera koji će biti prikazeni (strene 176).

Programskom linijom 10 odrađeno je da se unošenje obavlja od mamorijske lokacije sa adresom 30000. Svi primeri dati u ovom poglavlju će biti uneseni od te lokacije. Unošenje heksadecimalnih vrednosti se obavlja u rastućem poretku od nižih ka višim adresama, što u navedenom primaru znači s lava ne desno i odozgo na dole. Taj redosled ja najbolje ilustrovan u primeru broj 1.

Nakon unošenja dve haksadecimalne cifre, pritiskom na **ENTER** vrši sa upisivanja njihovog koda u memorijsku lokaciju. Završetak punjenja se obavlja pritiskom na tester **S** i **ENTER**.

Startovanje unesenog mašinskog koda (programa) ostvaruje se naredbom **USR** NN gde ja NN adresa od koja se žali izvršavanje mašinskog programa. Ta naredba sadržaj programskog brojača PC postavlja na datu vrednost NN, što omogućeva da procesor izvršava naredbe od adrese NN. Prethodna vrednost programskog brojača se čuva na mašinskom staku. odakle se uzima na kraju mašinskog programa kao povratna adresa u bejzik program. Najčešći oblici upotrebe naradbe startovanja mašinskog programa su sladeće.

1. **RANDOMIZE USR NN**. Naredbom **RANDOMIZE USR NN** se nakon završatka mašinskog programa postavlja nova početna vrednsot za izračunavanje slučajnog broja. Ta osnova ja sadržaj BC ragistar para koji je dobijen izvršavanjem mašinskog programa od adrese NN. Ovaj način pozivanja mašinskog programa se najčašće primenjuje, jer ne utiča na bejzik program iz koga je pozvan. Jedino će doći do uticaja na generisanje slučajnog broja.

2. **LET a = USR NN**. Izvršavanja mašinskog programe od adrese NN pomoću ove naredbe dovodi do toga da, nakon povratka u bejzik program, programska promenljiva **a** dobija vrednost sadržaja BC registar para. Ovaj način započinjanja izvršavanja mašinskog programa je primenljiv kada se mašinski program koristi za razna izračunavanja. Tada BC ragsitar par naposredno pre povratka u bejzik program treba napuniti rezultatom izračunavanja. Na taj način sa povratkom u bejzik program automatski izvršava dodela te vrednosti promenljivoj **a**.

3. **PRINT USR NN**. Kada se **USR** naredba primeni na ovaj način, pri povratku u bejzik program na ekranu če sa ispisati vrednost sadržaja BC registar para. Ova naredbe ča se primenjivati tokom datih primera jer se pomoću nja mogu efikasno prikazivati sadržaji ragistara procesora

Način pozivenja mašinskih iz bejzik programa nisu iscrpljeni kroz ova tri primera. Praktično se **USR NN** može upotrebiti u svim bejzik naredbama kao argument, ako je on predviđen.

Dati primeri mašinskog programiranja se izvršavaju do edrese 30000 pa se njihovo startovanje iz bejzik programe obavlja naredbom **RANDOMIZE USR 30000**. Biće posebno napomenuto kada je potrebno upotrebiti naredbu **PRINT USR** ili neku drugu.

PRIMER 1

Navedeni primer mašinskog programa ilustruje upotrebu naredbi osmobitnog i šesnaestobitnog punjenja registara konstantom, kao i punjenje memorijskih lokacija adresiranih registar parom

Program je dužine 7 bajta. Početak programa je na memorijskoj lokaciji 7530h (30000) i završava se na lokaciji 7536h. Unošenje kodova se može izvršiti nevedenim heksa punjaćem.

adresa	kôd	mnemonika
7530	3EFF	**LD A,FF**
7532	210040	**LD HL, 4000**
7535	77	**LD (HL), A**
7536	C9	**RET**

Prva naredba je naredba 8-bitnog punjenja u kojoj se registar A puni 8-bitnom konstantom. U primeru je ta konstanta FFh. Naredba je dužine dva bajte i unosi se u memorijske lokacije 7530h i 7531h. Druga naredba je naredba punjenja 16-bitnom konstantom registar para HL. Konstanta je u ovom slučaju 4000h. Viši, H registar se puni vrednošću 40h, a niži, L registar vrednošću 0. Ova naredba je trobajtne, kao i treća naredba punjenje memorijske lokacije adresirene HL registrom vrednošću sadržanom u A registru. U primeru se obavlja punjenje memorijske lokacije 4000h vrednošću FFh što odgovara bejzik naredbi **POKE 16384,255**. Četvrta naredba, na adresi 7536h, je naredba povratka na izvrševenje bejzik programe. Progremski brojač se puni edresom koja je bila sačuvana na steku prilikom odleska na izvršavanje mešinskog programa. Program se startuje bejzik naredbom **RANDOMIZE USR 30000**. Njegovim izvršavanjem se u memorijsku lokaciju 18384 (video memorija) upisuje vrednost 255 što će u gornjem levom uglu ekrana dati crticu dužine 8 tačaka.

Poznavanje mape memorije i organizacije video memorije je neophodno za potpuno razumevanje ovog i sledećih primera.

Uz izmenu sadržaja akumulatora može se odrediti koje će se tečke iscrtaveti. Sadržaj akummulatora se može izmeniti novim mašinskim kodom koji se najjdenostavnije upisuje bejzik naredbom **POKE** u memorijskoj lokeciji 7531h.

Promenom sadržaje HL registar para menja se položaj crte ne ekranu. Za vrednost sadržaja HL registar para od 58ØØh (22528) do 5BØØh (23296), što odgovara adresama polja atributa, mogu se postaviti odgoverajući atributi. Teda su boja, sjajnost i treptenje odgovarajućeg karaktera određeni sadržajem ekumulatora.

PRIMER 2

U ovom primeru su prikazene naredbe ze punjenje registara sadržajem memorijske lokecije i obrnuto, za punjenje memorijske lokacije sadržajem registra.

adrese	kôd	mnemonika
753Ø	3ADØ5C	**LD A, (5CDØ)**
7533	FD77FF	**LD (IY-1), A**
7536	C9	**RET**

Program je dužine 7 bejtova. Prva tri su kodovi za punjenje registra A sadržajem memorijske lokacije 5CDØh (2376Ø). Ta lokacija odgovara prvom znaku koji sledi iza prve naredbe u bejzik programu. Konkretno, u programskoj liniji **1 REM xyz** znak x se nalazi na toj lokaciji. Kôd slova x iz **REM** linije je prekopiren u registar A prvom naredbom, a drugom se tim kodom puni memorijska lokacija adresirane vrednošću (IY-1). Stalna vrsdnost sadržeja IY registra, prilikom izvršavanje bejzik programe, izabrana je da ukazuje ne sistemske promenljive i iznosi 5C3Ah (2361Ø). Memorijska lokacija sa edresom IY-1 je sistemska promenljiva PIP koja određuje trajanje tonskog odziva tastature. Izvršenjem gornjeg programa kôd znake x. 78h (12Ø), prebacuje ss u nju, uzrokujući odgovarejuće produženje tona. Promenom znaka x u neki drugi znek, uz ponovno izvršavanje programe, keo rezultat se dobija promena trajanja tonskog odzive testera.

Deti program, sem kao primer, nema veću upotrebnu vrednost. Njime se pokazuje mogućnost unošenja promenljivih u mešin-

ski program pomoću bejzik naredbi i programa, konkretno preko **REM** linije što se često sreće u programerskoj praksi. Napomena: Ako se gornji program unosi preko asemblera, treba proveriti da li asembler dozvoljava korišćenja naredbi koje konste IY registar par.

PRIMER 3

Mašinske naredbe punjenja ragistara sadržajem drugog registra i naradba zamene prikazane su u ovom primeru:

adresa	kôd	Mnemonika
7530	11E803	**LD DE,03E8**
7533	EB	**EX DE,HL**
7534	44	**LD B,H**
7535	4D	**LD C,L**
7536	C9	**RET**

Program je dužine 6 bajta. Prva tri bajta su kodovi 16-bitnog punjanja DE registar para konstantom 03E8h (1000). Četvrti bajt je kôd naradbe zamene sadržaja ragistar parova HL i DE. Treća naredba je sadržana u petom bajtu i puni registar B sadržajem registra H. Četvrta naredba (u šastom bajtu) puni registar C sadržajem registra L. U poslednjem bajtu je kôd naredbe za povratak u bejzik program.

Izvršavanje ovog mašinskog programa treba obaviti bejzik naradbom **PRINT USR 30000.** Na taj način će sadržaj BC registar para (1000) biti ispisan na akranu, što će biti i potvrda da je iz DE registar para sadržaj prebačen preko HL registar para u BC registar par. Slično će se dogoditi i ako se u prvoj neradbi umesto DE izvrši punjenje HL registar para.

PRIMER 4

Ovaj primer ilustruje primanu naredbe stavljanja podataka na stek **(PUSH)** i vraćanja podataka sa staka **(POP)**.

adraaa	kôd	mnamonika	komentar
7530	21D007	**LD HL,07D0**	Napuni HL par aa 2000 dac.
7533	E5	**PUSH HL**	Saćuvaj privremano na steku
7534	C1	**POP BC**	Vrati sa steka u BC reg. par
7535	C9	**RET**	Povratak u bejzik

Prvom naredbom se obavlja 16-bitno punjenje Hl registar para brojem Ø7DØh (2ØØØ). Drugom naredbom se sadržajem HL registar para pune dve memorijske lokacije u RAM-u koje su adresirane registrom SP. Trećom naredbom se BC registar par puni sadržajem tih memorijskih lokacija. Izvršavanjem ovog primera mašinskog programa naredbom **PRINT USR 3ØØØØ**, sadržaj BC registar para će biti ispisan na ekranu.

PRIMER 5

Kroz ovaj primer, tačnije grupu primera, biće prikazene aritmetičko-logike naredbe, 8-bitne i 16-bitne. Za prikazivenje efekata izvršavanja naredbi najpogodnije je dobijenim rezultatom napuniti BC registar par. Izvršavenje ovih primera treba obavljati bejzik naredbom **PRINT USR 3ØØØØ**.

OSMOBITNO SABIRANJE I ODUZIMANJE

adresa	kôd	mnemonika
753Ø	ØØ	**NOP**
7531	3Exx	**LD A,xx**
7533	Ø6yy	**LD B,yy**
7535	8Ø	**ADD A,B**
7536	Ø6ØØ	**LD B,Ø**
7538	4F	**LD C,A**
7539	C9	**RET**

Navedeni program demonstrite naredbu **ADD A,B**. Umesto xx i yy treba uneti vrednosti nad kojima se želi izvršiti operacija sabiranja. Te vrednosti će biti smeštene u lokacijama sa adresom 7532h (3ØØØ2) i 7534h (3ØØØ4).

U prvom bajtu ovog primera nalazi se kôd Ø koji odgovera naredbi kontrole procesora. Nailaskom na naredbu **NOP**, procesor jednostavno prelazi na izvršavenje naredne naredbe. **NOP** naredba je tu postavljena da bi se bejzik naredbom **POKE 3ØØØØ,55** zemenila sa **SCF** koja postavlja bit prenose na jedinicu. Uz dodatnu naredbu **POKE 3ØØØ5,136** zamenjuje se kôd naredbe **ADD A,B** u kôd neredbe **ADC A,B** što omogućava da dati program demonstrira i osmobitno sabiranje sa prenosom.

Isti program zamenom kodova na navedenim edresama kodovima naredbi **NOP** i **SUB B** (kôd **SUB B** je 9Øh = 144) prikazuje na-

redbu osmobitnog oduzimanja. Zamenom kodovima naredbi **SCF** i **SBC B** (kôd **SBC B** je 98h = 152) može se videti efekat naredbe oduzimanja sa prenosom.

Naredbe poredenja su naredbe fiktivnog oduzimanja što znači da se sadržaj A registra ne menja. Dolazi samo do izmena pokazatelja stanja. To će biti prikazano u narednim primerima.

Aritmetičke naredbe osmobitnog povećanja **INC** i smenjenje **DEC** su ostavljene čitaocu da ih sam upotrebi. Svekako da će u primerima koji slede biti upotrebljene i ove naredbe.

OSMOBITNE LOGIČKE NAREDBE

U ovoj grupi su naredbe **AND, OR** i **XOR.** Sledeći mašinski program prikazuje upotrebu ovih naredbi:

adresa	kôd	mnemonika
7530	3Exx	**LD A,xx**
7532	06yy	**LD B,yy**
7534	A0	**AND B**
7535	0600	**LD B,00**
7537	4F	**LD C,A**
7538	C9	**RET**

Umesto xx i yy treba uneti 8-bitne vrednosti nad kojima se želi izvršavanja programa naredbom **PRINT USR 30000** na ekranu će se ispisati rezultat.

Kao što je i ranije rečeno, za razumevanje logičkih operacija neophodno je predstavljanje brojeva koji učestvuju u operacijama u binarnom obliku. Mala vežba iz bejzik programiranja bi se sastojale u pisanju programa za pretvaranje heksadecimalnih, odnosno decimalnih brojeva u binarni oblik. Primer 9 pokazuje kako bi se to ostvarilo pisanjem programa u mašinskom jeziku.

Zamenom koda naredbe **AND** kodom naredbe **OR**, odnosno **XOR**, omogućuje se izvršavanje programa i dobijanje rezultata za naredbe logičkog sabiranja i logičkog isključivog sabiranja.

ŠESNESTOBITNO SABIRANJE I ODUZIMANJE

adresa	kôd	mnemonika
7530	01xxxx	**LD BC,xxxx**

7533	21yyyy	**LD HL,yyyy**
7536	09	**ADD HL,BC**
7537	44	**LD B,H**
7538	4D	**LD C,L**
7539	C9	**RET**

Navađeni program demonstrira naradbu **ADD HL,BC.** Umesto xxxx i yyyy traba unati 16-bitna vrednosti nad kojima se želi izvršiti operacija sabiranja. Isto kao i u primeru 8-bitnog sabiranja u prvom bajtu programa je sadržana naredba **NOP.** Nju treba zamaniti naredbom postavljanja bita prenosa na jadinicu **(SCF)** ukoliko se žale prikazati naredba 16-bitnog sabiranja i oduzimanja sa prenosom.

PRIMER 6

U ovom primeru su prikazane naredbe skoka, poređanja, povećavanja i smanjenja za jadan.

adrasa	kôd	mnemonika
7530	0600	**LD B,00**
7532	3E05	**LD A,05**
7534	FE05	**CP 05**
7536	2803	**JR Z,L1**
7538	3D	**DEC A**
7539	4F	**LD C,A**
753A	C9	**RET**
753B	3C	**L1 INC A**
753C	4F	**LD C,A**
753D	C9	**RET**

U prvoj naredbi se B registar puni vrednošću 0 da bi se obazbedilo ispisivanje sadržaja samo C ragistra. U drugoj naredbi sa A registar puni vrednošću 5. Treća naredba je naredba poređenja sadržaja akumulatora sa vrednošću 5. Kako je već rečeno, naredba poređenja je po efektu slična naredbi oduzimanja, s tim što se sadržaj akumulatora ne menja. Biti pokazatelji stanja se postavljaju isto kao i u slućaju naredbe oduzimanja. Za vrednosti navedene u primeru, rezltat oduzimanja je nula što će postaviti pokazatelj nule (Z bit) na jedinicu.

Četvrta naredba je naredba relativnog skoka u slučaju da je pokazatalj nule jednak jedinici. S obzirom da je taj uslov ispunjen, program će nastaviti sa izvršavanjem na adresi memorijske lokacije koja ja za 3 veća nego u slučaju da uslov nije bio ispunjan. To znači da sa izvršavanjem sledeća naradbe povećava sadržej A registre za jadan **(INC A)**. To će biti potvrđano izvršavanjem programa naredbom **PRINT USR 30000** i ispisivanjem rezultata koji je jadnak 6.

Ukoliko se ragistar A u drugoj naradbi napuni nakom drugom vradnošću, naće doći do ispunjenja uslova potrebnog za skok. U tom slučaju ća program nastaviti da se izvršava od naredbe **DEC A,** što će za razultat dati broj 4.

U osmoj naredbi, **INC A**, memorijskoj lokaciji 753Bh pridodato je ime, labela L1. Četvrta naredba se poziva na labelu L1, odnosno na njenu memorijsku lokaciju. Označavanja labelama, iako ne predstavlja standardnu mnemoniku procesora Z80, nezamenljivi ja deo asamblarskog programa.

Na osnovu ovog primera i svega što je do sada izloženo mogu se baz većih teškoća izvršiti izmene za prikazivanje ostalih naredbi skoka.

PRIMER 7

Ovaj primer prikazuje upotrabu naredba skoka koji ja izvršen ako je sadržaj B ragistra različit od nule **(DJNZ).**

adresa	kôd	mnemonika
7530	210040	**LD HL,4000**
7533	3EFF	**LD A,FF**
7535	0620	**LD B,20**
7537	77	**L1 LD (HL),A**
7538	23	**INC HL**
7539	10FC	**DJNZ L1**
753B	C9	**RET**

Prvom narebom sa HL registar par puni adrasom prvog bajta video memorija. Drugom i trećom naredbom sa ragistri A i B pune vrednostima FFh i 20h. Četvrta naredba puni prvu mamorijsku lokaciju vidao mamorije sadržajem A registra. Peta naradba povećava za jedan sadržaj HL ragistar para ukazuju na novu lokaciju u memoriji. Sledaća naradba ja naredbe skoke uz uslov da sadržaj B regis-

tra ima vrednost različitu od nule. Taj uslov je u prvom prolasku ispunjen, što će za datu vrednost operanda u naredbi skoka dovasti do skoka na memorijsku lokaciju čija je adresa za četiri manja. Naredbom **DJNZ** se istovremeno vrši smanjivanje vrednosti sadržane u registru B.

Program ća sa nastaviti naradbom **LD (HL),A** što ća dovesti do upisivanja vradnosti FFh u novu, susednu, memorijsku lokaciju. Cao ciilus ća se ponoviti 32 puta što će kao rezultat na ekranu dati liniju dužine 32 karaktera.

Izvršavanja ovog programa je adakvatno obaviti naradbom **RANDOMIZE USR 30000.**

PRIMER 8

Upotraba naradba poziva potprogrgama je prikazana u ovom primeru:

adresa	kôd	mnamonika	komentar
7530	3E01	**LD A,01**	Napuni A ragistar sa 1
7532	CD3C75	**CALL L1**	Idi na potprogram L1
7535	CD3C75	**CALL L1**	Idi ponovo na isti potprogram
7538	0600	**LD B,00**	Obezbedi štampanje samo C registra
753A	4F	**LD C,A**	Obezbadi štampanja sadržaja A registra
753B	C9	**RET**	Povratak
753C	3C	**L1 INC A**	Potprogram, povećaj sadržaj A registra
753D	C9	**RET**	Povratak u glavni program

U glavnom programu sa potprogram poziva dva puta i svaki put sa sadržaj A registra povaćava za jedan. Na ekranu ća se ispisati vrednost sadržaja akumulatora, što je u ovom slućaju 3, jar sa vrednost 1 dva puta povećava za jadan.

Ostala naradba poziva potprogrâma zavisa od sadržaja u ragistru pokazatelja stanja i mogu sa prikazati kratkim programima. Primere upotrebe naredbi uslovnog povratka treba da uradi sam čitalac.

PRIMER 9

U ovom progremu će biti prikazano keko se primenjuju naredbe rotacije i restert naredbe.

Peta naredba u programu je naredba resterta, odnosno odlaska na potprogram koji počinje na lokaciji sa adresom 10h. To je adresa u ROM-u ne kojoj se nalazi ROM rutina za ispisivenje sadržaja A registra.

adresa	kôd	mnemonika	komentar
7530	16xx	**LD D,xx**	xx-željena 8-bitna vrednost
7532	0608	**LD B,08**	Za ponavljanje 8 puta
7534	3E30	**L2 LD A,30**	30h – kôd cifre 0
7536	CB22	**SLA d**	Za bit ulevo sedržaj D reg.
7538	3001	**JR NC,L1**	Skok ako nema prenosa
753A	3C	**INC A**	Ime. 31h-kôd cifre 1
753B	D7	**L1 RST 10**	Ispiši sedržaj A reg.
753C	10F6	**DJNZ L2**	8 puta
753E	C9	**RET**	Povratak

Program ispisuje sadržaj D registra u binernom obliku. Četvrtom naredbom se obavlja pomerenje sadržaja D registra za jedno mesto ulevo. Krajnji levi bit pomeranjem prelazi u bit prenose definišući uslov za izvođenje naredbe skoka. U slučeju da je bit prenosa postavljen na jedinicu, neće doći do skoka, e vrednost sadržaja registra A će se povećati za jedan. Novi sadržaj registra A je teda 31h, što je kôd broja 1. Na ekránu će se ispisati 1. Za slučej da je ispunjen uslov skoka, preskače se naredba za povećanje sadržaja A registra ze 1 i na ekranu se ispisuje 0. Postupak se ponavlja 8 puta ostvarujući ispisivanje 8-bitnog broja u binarnom obliku.

Izvršavanje ovog programe se zbog upotrebe naredbe **RST 10** izvodi bejzik naredbom

PRINT: RANDOMIZE USR 30000

Veliki broj naredbi rotacije onemogućava da se sve prikažu. Ukoliko je čitalac pažljivo proučio prethodni primer, neće biti teško de se dode do potpunog ovladavanja i primenjivanja naredbi rotacija.

PRIMER 10

U ovom primeru je demonstrirana upotreba neredbe iz grupe za manipulaciju bitima keo i naredba uslovnog povratke.

adresa	kôd	mnemonika	komentar
7530	010000	**LD BC,0**	Sadržaj BC para postviti ne 0
7533	21xxxx	**LD HL,xxxx**	xxxx-adrese željenog bajte
7536	CB46	**BIT 0 ,(HL)**	Bit 0, da li je nula (paran broj)
7538	C0	**RET NZ**	Nije. Povratak uz BC = 0
7539	03	**INC BC**	Jeste. Povećanje i povratak uz BC = 1
753A	C9	**RET**	

Program ne osnovu vrednosti bita najmanje težine u adresiranoj memorijskoj lokeciji utvrduje da li je vrednost njenog sedržaja paran ili neperan broj. Za nepernu vrednost na ekranu će se ispisati broj 0, a za parnu 1.

Izvršenje programa se obevlja uobičejenom naredbom **PRINT USR 30000**

PRIMER 11

Naredbe za prebacivanje blokova prikazane su u ovom primeru:

adresa	kôd	mnemonika	komentar
7530	210040	**LD HL,4000**	Od memorijske lokacije 4000h
7533	110050	**LD DE, 5000**	Na memorijsku lokeciju 5000h
7536	010008	**LD BC,0800**	U dužini 0800h
7539	EDB0	**LDIR**	Prebeci sadržaj memorijskih lokacija
753B	C9	**RET**	Povratek

Izvršavanjem programa naredbom **RANDOMIZE USR 30000: PAUSE 0** vrši se prebacivanje sadržaja gornje trećine ekrana u donju trećinu. Sedržaj memorijskih lokacije sa adresema od 4000h do

4800h prebacuja se u memorijska lokacije sa adresama od 5000h do 5800h. Umesto naredbe **LDIR**, uz manje izmene u programu, mogu sa upotrebiti i ostale naredbe prebacivanja **LDDR, LDI i LDD.**

PRIMER 12

Pratraživanja blokove ja prikazano kroz naredbu poređenja, povećanja i ponavljanja.

adrasa	kôd	mnemonika	komentar
7530	3Exx	**LD A,xx**	xx-vrednost koja se traži
7532	210040	**LD HL,4000**	Od prva lokacija video memorije
7535	010018	**LD BC, 1800**	Svih 6144 lokacija vidao memorije
7538	EDB1	**CPIR**	Uporaduj I povećavaj sadržaj HL
753A	2B	**DEC HL**	Umanjenje; HL ukazuja za jadan više
753B	44	**LD B,H**	Prabacivanja u BC par
753C	4D	**LD C,L**	redi ispisivanja
753D	C9	**RET**	Povratak

Naredbom: **10 CLS: POKE 16384+INT(RND*6144),xx:PRINT USR 30000**

gde je xx ista osmobitna vradnost uneta i u prvu naradbu mašinskog programa, dobija se adresa lokacije u video memoriji koja je slučajnim izborom napunjena vrednošću xx. U slučaju napostojanja tražene vrednost HL registar par će ukazivati na poslednji bajt oblasti pretraživanja (22527).

PRIMER 13

Kroz sledeći primar biće prikazane ulazno-izlazne naredbe procasora Z80. U Spektrumu je procesoru Z80 periferna jedinica ULA kolo. Njana edresa je FEh (254).Treba uočiti da adrese perifernih jedinica nisu isto što i adrese memorijskih lokacija. Preko ULA procesor prima podatke sa EAR priključka i podatke o pritisnutosti

tastera. Takođe preko ULA sa određuje boje oboda akrana, aktivira zujalica i šalju podaci na MIC priključak.

U narednom primeru će biti prikazano kako se direktno očitava tastatura.

adresa	kôd	mnemonika	komantar
7530	3EFE	**LD A,FE**	Leva polovina donjeg reda tastera
7532	DBFE	**IN A,(FE)**	U/I jedinica FEh (ULA kolo)
7534	E61F	**AND 1F**	Maskiranje 5,6 i 7-og bita
7536	0600	**LD B,00**	Obezbeđivanja ispisivanje sadržaja A reg.
7538	4F	**LD C,A**	
7539	C9	**RET**	

Izvršavanjem druge naredbe u pet nižih bita ekumulatora postavljana je vrednost kojom sa ispituju tasteri donjeg reda (od CAPS SHIFT do V). Ispitivanja se vrši navedenim mešinskim programom. Startovanje programa i ispisivanje rezultate se obavlja bejzik programskom linijom.

10 PRINT AT 5,5; USR 30000: GO TO 10

U slučeju da ni jeden tester nije pritisnut ispisaće se vrednost 31. Pritiskanjem jednog ili više tastere istovremeno mogu se uočiti „težine tastera" i potrebne zakonitosti za direktno mašinsko dekodovanje tastature.

S obzirom da je prvih pet bita od b0 do b4 predviđeno ze dekodovanje tastature, a b5 za signel iz EAR priključke, moguće je očitavanje samo po pet tastera iz jednog reda, tj. polurede. Očitavanje ostalih poluredove se dobija zamenom vrednosti kojom se puni A register u prvoj naredbi. Te vrednosti su: FEh (254) za levu polovinu donjeg reda, FDh (253) za levu polovinu drugog reda, FBh (251) za levu polovinu trećeg rede, F7h (247) za levu polovinu gornjeg rada dok je EFh (239) za desnu polovinu. Za desnu polovinu trećeg reda je DFh (223), drugog rede BFh (191) i donjeg rede 7Fh (127). Jasniji prikaz navedenih vrednosti je dat tabelom 4-19.

Na ovaj način ja olakšano nalaženje potrebna vrednosti za punjenje akumulatora pri istovremenom dekodovanju više poluredova (npr. za leva dva donja polureda ta vrednost iznosi 255-3, tj. 252).

1	255 – 8	255 – 16	Ø
Q	255 – 4	255 – 32	P
A	255 – 2	255 – 64	ENTER
C.S.	255 – 1	255 – 128	BREAK

Tabela 4-19 Vrednsoti dodeljene poluredovima tastature

Izlazne naredbe procesora Z80 se mogu izvesti adekvatno ulaznim neredbama. Čitaocu se ostavlja da to sam učini za slučej ULA kola. Adresiranje ULA kola se obavlja sa nevedenom vrednošću FEh ne nižih osam bita adresne magistrale (bØ-b7).

Oblik ove naredbe če biti

OUT A,(FE)

Sedržaj ekumulatora če odrediti stanje na pojedinim izlaznim jedinicama ne sledeći način: biti bØ, b1 i b2 određuju boju obodnog dela ekrane (border), b3 određuje stanje na izlazu MIC, a b6 stanje na zvučniku.

Kroz prethodne primere neposredno su prikazane naredbe procesora Z80. Veliki broj naredbi i načine njihovog korišćenja ne omogućuje njihovo celokupno prikazivanje odgovarajućim primerima. Izborom prethodnih primera nastojalo se prikazati što šire područje upotrebe. Kroz razumevenje ovih primera čitaocu če biti omogućeno da samostelno primeni ostele neredbe procesora.

Naredni primeri treba da potvrde da se na osnovu svega što je do sade naučeno kroz programiranje u bejziku I kroz upotrebu mašinskih naredbi mogu formirati nemenski mašinski programi.

PRIMER 14

U ovom primeru se vrši komplementiranje sadržaja video memorije. Poznato je de biti postavljeni na jedinicu odgovaraju zapisu na ekranu. Invertovanjem nula i jedinice u celokupnom sadržaju video memorije, na ekranu se dobije invertovana slika. Treba uočiti da je efeket isti keo i pri upotrebi bejzik naredbi za zamenu vrednosti boje zapisa **(INK)** i boje osnove **(PAPER)**. Razlika je u tome što se bejzik naredbom to ostvaruje u polju atributa, a ovim progremom u video memoriji.

adresa	kôd	mnamonika	komentar
7530	210040	**LD HL,4000**	HL = prvi bajt vidao memorije
7533	3EFF	**L1 LD A, FF**	Priprema A registra
7535	AE	**XOR (HL)**	Dobijanje komplementa
7536	77	**LD (HL),A**	Punjenje vidao memorije
7537	23	**INC HL**	Prelazak na sledeći bajt
7538	110058	**LD DE,5800**	DE = Poslednji bajt vidao memorije
753B	D3FE	**OUT (FE),A**	Signal ka zujalici, obodu, MIC
753D	B7	**OR A**	Postavljanje C bita na 0
753E	E5	**PUSH HL**	Privremeno sačuvati HL
753F	ED52	**SBC HL,DE**	Oduzeti trenutnu adresu od poslednje
7541	E1	**POP HL**	Povratiti trenutnu adresu u HL
7542	38EF	**JR C,L1**	Ako nije poslednja, skok na L1
7544	C9	**RET**	Povratak

Aktiviranje zujalice i promena boje oboda akrana se obavlja izlaznom naredbom zavisno od sadržaja video memorije.

PRIMER 15

Navedeni program levi deo slike na ekranu prebacuje na dasnu stranu i obrnuto čime se dobija „slika u ogledalu". To se ostvaruje međusobnom zamenom bitova koji odgovaraju levoj i desnoj strani slike u odnosu na vertikalnu osu za svih 192 redova. Zamena se obavlja u grupama od po 8 bita. Memorijska lokacija 7568h se koristi za privremeno čuvanje prathodne vrednosti jednog bajta lokacije u vidao memoriji. Poslednja naredba u programu kojoj je dodeljena labela XX definiša 1 bajt u memoriji predviđen za to. Iako se na radi o standardnoj mnemoničkoj oznaci **DEFB N** za procesor Z80 ona je konvencionalno asemblerska naredba koja memorijskoj lokaciji dodeljuje sadržaj N.

adresa	kôd	mnemonika	komentar
7530	06BF	LD B,C0	Broj karakter redova ekrana
7532	11E03F	LD DE, 3FE0	Adresa prvog bajta – 32
7535	C5	L2 PUSH BC	Sačuvati broj 8
7536	212000	LD HL,20	Dodatak za svaki novi red
7539	19	ADD HL,DE	U HL je adresa prvog bajta reda
753A	EB	EX DE, HL	U DE ja adresa prvog (levog) bajta
753B	211F00	LD HL, 1F	Dodavanja 31, dobija se
753E	19	ADD HL,DE	u HL adresa desnog bajta u radu
753F	D5	PUSH DE	Sačuvati obe adrese
7540	E5	PUSH HL	krajnje levog i krajnje desnog bajta
7541	0610	LD B,10	U redu je 16 takvih parova
7543	C5	L1 PUSH BC	Sačuvaj i to
7544	1A	LD A,(DE)	U A reg. vrednost levog bajta
7545	326875	LD (XX),A	To sačuvati u mem. lokaciji XX(7568h)
7548	7E	LD A,(HL)	U a reg. vrednost desnog bajta
7549	CD5F75	CALL POD	Odlazak na potprogram (755Fh)
754C	12	LD (DE),A	U levu memorijsku lokaciju staviti novu vrednost
754D	3A6875	LD A,(XX)	U A registar stviti vradnost levog bajta
7550	CD5F75	CALL POD	Odlazak na potprogram (755Fh)
7553	77	LD (HL),A	U desnu mem. lok. staviti novu vrednost
7554	13	INC DE	Prelazak na susadni par
7555	2B	DEC HL	–

7556	C1	**POP BC**	Povratiti u B broj perova
7557	10 EA	**DJNZ L1**	Ukupno 16 puta
7559	E1	**POP HL**	Povratiti obe početne edrese
755A	D1	**POP DE**	–
755B	C1	**POP BC**	Povratiti broj redova
755C	10 D7	**DJNZ L2**	Ukupno 192 puta
755E	C9	**RET**	Povratak u bejzik
755F	06 08	**POD LD B,08**	B = broj bitova u bajtu
7561	17	**LP RLA**	Rotacije ulevo levog bejta
7562	CB19	**PR C**	C reg. se puni bitom prenosa
7564	10 FB	**DJNZ LP**	Osam pute
7566	79	**LD A,C**	Rezultat iz C reg. staviti u A reg.
7567	C9	**RET**	Povratak u glavni program
7568	00	**XX DEFB 0**	Ovde se čuva vrednost levog bajta

PRIMER 16

U ovom primeru se prikazuje izvršavanje programa kada procesor sa prekidom radi ne način 2 (mode 2).

NAPOMENA: Zbog greške u konstrukciji Spektruma to se ne može ostvariti uspešno u verzijama od 16 Kbajta RAM-a.

Za upotrebu navedenog programe neophodno je prethodno uneti već dat program ze invertovanje slike u primeru 14. Taj program se nelazi u memorijskim lokacijama sa adresama od 7530h do 7544h. U lokacije od adrese 7545h do 7559h unosi se dole navedeni program. Drugi deo programa se unoei od memonjske lokacije F4FFh (62719), što zahteve izmenu u liniji 10 programa heksa punjača **(LET adr = 62719)**. Program se unosi do lokacije F50Eh.

adrese	kôd	mnemonika	komentar
7545	F3	**DI**	Onemogućiti na dalje prekid
7546	F5	**PUSH AF**	Sačuveti A i F registre
7547	3EFD	**LD A,253**	Polured testature od A do G

7549	DBFE	**IN A,(FE)**	Učitati
754B	1F	**RRA**	Bit nejmanje težine?
754C	3807	**JR C,IZLAZ**	Jedinica. Taster A nije pritisnut
754E	D5	**PUSH DE**	Nula. Sačuvati vrednost registra
754F	E5	**PUSH HL**	Koji će se koristiti, DE i HL
7550	CD3075	**CALL 7530**	Odlazak na potprogram
7553	E1	**POP HL**	Vretiti sačuvane vrednosti HL i DE
7554	D1	**POP DE**	–
7555	F1 **IZLAZ**	**POP AF**	Vratiti vrednost u A i F registre
7556	FF	**RST 38**	Očitati testeturu
7557	FB	**EI**	Dozvoliti ponovo prekid
7558	ED4D	**RETI**	Povratak iz prekidne rutine
F4FF	45	**DEFB 45**	7545h – adresa prekidne servisne
F500	75	**DEFB 75**	rutine
F501	3E3F	**LD A,3F**	Isključenje prekidne servisne
F503	ED47	**LD I,A**	rutine
F505	ED56	**IM 1**	
F507	C9	**RET**	
F508	3EF4	**LD A,F4**	Uključenje prekidne servisne rutine
F50A	ED47	**LD I,A**	
F50C	ED5E	**IM 2**	
F50E	C9	**RET**	

Način rada dve aktivira se naredbom **RANDOMIZE USR 62728,** a deaktivira naredbom **RANDOMIZE USR 62721.** Nailaskom signele prekida koji šalje ULA kolo 50 puta u sekundi, procesor prelazi na izvršavanje programa, servisne rutine, od memorijske lokacije indirektno adresirane sadržajem I registra na viših 8 bita adresne magistrale, i sa FFh (u slučaju ULA kole) na nižih 8 bita adresne ma-

gistrala. U datom primeru to daje mamorijsku lokaciju F4FFh na kojoj se nalazi adrasa 7545h odakla počinje servisna rutina.

Sarvisna rutina se sastoji iz ulaznog, radnog i izlaznog dela. U uzlaznom dalu se uobičajano vrši provera dozvola ulaska u radni dao i čuvanje tekućih vrednosti registara procesora i dala RAM-a koji se koriste od strana servisne rutine. Radni dao neposredno izvršava program pradviđen za izvršavanja pri pojavi signala prekida. Izlazni deo obezbađuje nastavljanja programa koji ja bio prekinut. U datom primeru radni deo servisne rutine je od lokacije 7530h do 7544h, a ulazni i izlazni dao od lokacije 7545h do 7559h.

Aktiviranjam načina dva, navedeni program obazbeđuje da na svaki pritisak tastera A dolazi do zaustavljanja takućeg programa uz izvršavanje predviđene servisne rutine invertovanja sadržaja ekrana. To se može demonstrirati pritiskanjem tastera A u toku izvršavanja naredba

FOR N=1 TO 1000: PRINT N;:NEXT N

Način 2 rada procesora pruža velike mogućnosti u rešavanju specifičnih problema koji se mogu pojaviti u praksi, ali programer mora pravilno da odgovori na pitanje opravdanosti uvođenja prakida za njihovo rešavanje.

```
  10 LET adr=30000: POKE 23658,8
  20 LET w$="": INPUT "Kod",h$:
IF h$="S" THEN  STOP
  30 IF LEN h$<>2 THEN  GO TO 20
  40 LET kod=16*(CODE h$-48-7*(C
ODE h$>64))+(CODE h$(2)-48-7*(CO
DE h$(2)>64)): POKE adr,kod
  50 LET a3=INT (adr/4096): LET
a2=INT ((adr-4096*a3)/256): LET
a1=INT ((adr-4096*a3-256*a2)/16)
: LET a0=adr-4096*a3-256*a2-16*a
1
  60 LET a=a3: GO SUB 100: LET a
=a2: GO SUB 100: LET a=a1: GO SU
B 100: LET a=a0: GO SUB 100
  70 PRINT w$,h$: LET adr=adr+1:
 GO TO 20
 100 LET a=a+48+7*(a>=10): LET w
$=w$+CHR$ a: RETURN
```

Program punjač (LOADER)

ROM i upotreba ROM rutina

5

Celokupan rad Spektruma se zasniva na izvršavanju 16 Kbajta mašinskog programa koji se nalazi u ROM-u. Izuzetak je izvršavanje mašinskih programa koje je korisnik uneo u računar. Mašinski program u ROM-u se naziva monitorski program (operativni sistem i bejzik interpretar) i odlikuju se velikom kompleksnošću. Razumevanje monitorskog programa predstavlja put ka razumevanju rada Spektruma i rada drugih računara.

U ovom poglavlju ča biti dat kraći pregled monitorskog programa, a kompletan disasemblerski prikaz, koji je nezamanljiv pri potpunom upoznavanju sa računarom, može sa naći u navedenoj literaturi pod brojem 3.

Monitorski program se sastoji iz više potprograma–rutina Glavne rutina monitorskog programa su:

Rutina inicijalizacija: Počinje od adrese Ø i u potpunosti se izvršava od tranutka uključenja računara do ispisivanja poruke o autorskom pravu Njenim izvršavanjem se obavlja provera raspoložive memorije, postavljaju se radni prostori i sistamske promenljive.

Glavna izvršna rutina: To je najvažnija rutina monitorskog programa Sastoji se iz petlje koja omogučava formiranje bejzik linija i bejzik programa. U njoj se vrši provara sintakse upisane linije i samo ako su pravila formiranja linije zadovoljana, doći če do njenog smeštanja u deo memorije u kojoj se formira bejzik program U diraktnom načinu rada bejzik linija se predaje interpreteru na izvršavanje. Posredstvom glavna rutine če se takode uputiti odgovarajući izveštaj.

Editoraka rutina: Editor omogućuje korisniku računara ispisivanje bejzik linija. Program koji se unosi preko tastature u obliku kôda formira se u editorskom delu RAM-a ispisivanjem programske linije u donjem delu ekrana. U zavisnosti od toga da li se obavlja unošenje karaktera tj. naredbi ili editorskih komandi (kontrola pokazivača), vrši sa pozivanje odgovarajučih rutina editora. Pritiskom na taster ENTER prelazi sa u glavnu izvršnu rutinu koja če preuzeti

unesene karaktere. Editorska rutina se aktivira i pri unošenju karaktera pomoću bejzik **INPUT** naredbe.

Rutina dekodovanja tastature: Aktivira se 50 puta u sekundi pri svakom signalu prakida koji ULA šalje procesoru i proverava pritisnutost tastera. Kôd pritisnutog tastera se smašta u sistemsku promenljivu LAST K (23560). Pati bit promenljive FLAGS (23611) sa postsvljs na jedinicu da bi se potvrdilo da je pritisnut novi taster.

Rutina ispisivanja karaktera: Ove rutina predstavlja vrlo značajan dao ROM-a. Svako ispisivanje karaktara na ekranu ili štampaču obavlja se upotrebom ova rutine. Rutina se poziva preko memorijske lokacija sa adresom 10h (RST 10h). Rutina u toku svog izvršavanja poziva podrutine koje za zadate kodove nalaze odgovsrajuća karaktera u karakter setu ROM-a ili UDG polja u RAM-u. Kodovi se preuzimeju i prebacuju u vidao memoriju, što dovodi do njihovog prikazivenja na ekranu.

Bejzik interpreter: Obezbeduje izvršavanje bejzik programskih linija njihovim prevodenjem. Pri tome izvršava i proveru sintakse. Za izvršavanje svih 50 bejzik naredbi koristi se 50 komandnih rutina. Potrebna komandna rutina se nalazi utvrđivanjem klase kojoj naredbi pripada i na osnovu komandne tabale koja sadrži adresa svih 50 naredbi.

Rutina za računanje (kalkulator): Veliki deo monitorskog programa čine složene rutine za decimalna izračunavanja. I pored njene kompleksnosti, sistematizacijom je omogućana vrlo praktična upoteba računarskih podrutina.

Karakter set: Nalazi se od memorijske lokacije 3D00h. Sastoji se od 96 karaktera. Svaki karakter je predstavljen sa 64 bita u organizaciji datoj u prikazu organizecije video memorije. Slovo A koje se nalazi od adrese 3D00h prikazano je sledećim vrednostima bajta.

Adresa	Kôd	Binarni oblik
3D00	00	00000000
3D01	3C	00111100
3D02	42	01000010
3D03	42	01000010
3D04	7E	01111110
3D05	42	01000010
3D06	42	01000010
3D07	00	00000000

U 16 Kbajta ROM-a sa od adrese 386Eh do 3CFFh nalazi takođe i 1170 neupotrebljenih memorijskih lokacija.

Monitorski program ROM-a sa navadenim glavnim ROM rutiname sastoji se iz velikog broje podrutina koje korisnik može pozivati i koristiti u svojim bajzik ili mašinskim programima. U navedenom tekstu daće se prikaz najzanimljivijih ROM rutina i način njihovog korišćanja.

5-1 RUTINE EKRANA

ROM rutine akrana omogućuju brisanje i pomeranja sadržaja akrana. Pre nego što se pogledaju najzanimljivija rutine koje to ostvaruju, neophodno je reći da je broj od 24 reda na ekranu konstantan, a da podaia na 22 reda u gornjem delu i dva editorska reda u donjem daiu ekrana varira u toku rada računara. Za oba dela se koriste iste rutine s tim što ja za izlazak na svaki dao ekrana potrebno aktivirati odgovarajući upravljački (kanalski) program, odnosno izvršiti otvaranja odgovarjaućag kanaia.

RUTINE BRISANJA EKRANA

Rutina brisanja sadržaja ekrana u Spaktrumu omogućuju brisanje celog ekrana i brisanje donjeg dela ekrana. Brisanje celog ekrane i postavljanje atributa za gornji dao ekrana na vradnost stalnih boja (sistemska promenljiva ATTR–P) ostvaruje se pozivanjem rutine na adresi 0D6Bh nakon otvaranja kanala S.

adresa	kôd	mnemonika	komentar
7530	3E02	**LD A,02**	Otvoriti kanal S
7532	CD0116	**CALL 1601**	ROM rutina CHAN–OPEN
7535	CD6B0D	**CALL 0D6B**	Poziv ROM rutine CLS
7538	C9	**RET**	Povratak

Rutina brisanja sadržaje donjeg dela ekrana omogućuje brisanje donjag dela akrana. Vrednost sadržaja B registre odgovara broju obrisanih linija pri brojanju odozdo na gora.

adresa	kôd	mnemonika	komentar
7530	0605	**LD B,05**	B = broj linija donjeg dela ekrana

7532	CD440E	**CALL 0E44**	ROM rutina CL-LINE
7535	C9	**RET**	Povratak

RUTINA POMERANJA SADRŽAJA EKRANA

Ova rutina se primanjuje u Spektrumu pri pomeranju za jedan red naviše. Njena primena se svodi na naredbu **RANDOMIZE USR 3190** (decimalno) pri upotrebi u bejzik programima, odnosno **CALL 3190** u mašinskim programima.

5-2 RUTINE ISPISIVANJA

Rutine ispisivanja novih sadržaja ekrana, PRINT rutine, omogućuju ispisivanje pojedinih karaktera i nizova karaktera.

RUTINA ISPISIVANJA KARAKTERA je glavna rutina ispisivanja u Spektrumu. Poziva se naredbom **RST 10h** kojom sa izvršava apsolutni skok na adresu 15F2h odakle počinje rutina ispisivanje karaktera. Ova rutina ima velike mogućnosti. Omogućuje ispisivanje svih karaktera, naziva naredbi, a takođe prihvata i kontrolne karaktere. Upotreba rutine ispisivanja karaktera se sastoji u sledećem: Potrebno je otvoriti određen kanal za mesto na kome se želi ispisati karakter. To se ostvaruje za gornji deo ekrana punjenjem akumulatora brojem 2 i pozivanjem rutine za otvaranje kanala koja se nalazi na adresi 1601h. Za donji deo ekrana otvara se kanal K (registar A se puni brojem 0). Potrebno je otvoriti kanal P (akumulator napuniti brojem 3) da bi se obezbedilo ispisivanje na štampaču. Posle otvaranja kanala akumulator se puni kôdom karaktera koji se želi ispisati. Zatim se poziva rutina ispisivanja. Ponovnim punjenjem akumulatora novim kôdom i pozivanjem rutine naredbom **RST 10h**, vrši se ispisivanje novog karaktera.

Sledeći primer koji odgovara bejzik naredbi PRINT AT 5,10; PAPER 5; „MIKRO 85" ilustruje postupak ispisivanja upotrebom ROM rutine.

adresa	kôd	mnemonika	komentar
7530	3E02	**LD A,2**	Gornji deo ekrana
7532	CD0116	**CALL 1601**	ROM rutina CHAN--OPEN
7535	3E16	**LD A,16**	Kod AT kontrola

7537	D7	**RST 10**	ROM rutina PRINT A-1
7538	3E05	**LD A,05**	x koordinata
753A	D7	**RST 10**	
753B	3E0A	**LD A,0A**	y koordinata
753D	D7	**RST 10**	
753E	3E11	**LD A,11**	Kôd 'PAPER' kontrole
7540	D7	**RST 10**	
7541	3E05	**LD A,05**	Kôd svetloplava boje
7543	D7	**RST 10**	
7544	3E4D	**LD A,4D**	Kôd karaktera 'M'
7546	D7	**RST 10**	
7547	3E69	**LD A,69**	Kôd karaktera 'i'
7549	D7	**RST 10**	
754A	3E6B	**LD A,6B**	Kôd karaktera 'k'
754C	D7	**RST 10**	
754D	3E72	**LD A,72**	Kôd karaktera 'r'
754F	D7	**RST 10**	
7550	3E6F	**LD A,6F**	Kôd karaktera 'o'
7552	D7	**RST 10**	
7553	3E20	**LD A,20**	Kôd karaktera ' '
7555	D7	**RST 10**	
7556	3E38	**LD A,38**	Kôd karaktera '8'
7558	D7	**RST 10**	
7559	3E35	**LD A,35**	Kôd karaktera '5'
755B	D7	**RST 10**	
755C	C9	**RET**	

RUTINA ISPISIVANJA STRINGOVA omogućuju ispisivanje niza kodova u obliku odgovarajućih karaktera i naredbi. Pre pozivanja rutina preko adresa 203Ch potrebno je u DE registar par uneti adresu na kojoj sa nalazi prvi karakter niza. BC registar par treba napuniti vrednošću koja odgovara broju karaktera u nizu. Na poruci iz prethodnog primara prikazaće se upotreba rutine ispisivanja stringova.

adresa	kôd	mnemonika	komentar
7530	3E02	**LD A,02**	
7532	CD0116	**CALL 1601**	
7535	113F75	**LD DE,753F**	
7538	010D00	**LD BC,0D**	
753B	CD3C20	**CALL 203C**	ROM rutina PR--STRING
753E	C9	**RET**	
753F	16050A	**DEFB 16,05,0A**	
7542	1105	**DEFB 11,05**	
7544	4D696B72	**DEFB "M", "I", "k", "r", "o"**	
7549	20	**DEFB " "**	
754A	3835	**DEFB "8", "5"**	

S obzirom da rutina preko naredba CALL 203Ch koristi i naredbu RST 10h, string može da sadrži i kontroine kodova ispisivanja, a takode i kodove naredbi.

RUTINA ISPISIVANJA PORUKE predstavlja još jadnu rutinu za ispisivanje stringova. U Spektrumu se primenjuje za ispisivanje izvaštaja (raporta) i naradbi. Poziva se preko adrese 0C0Ah. Pre pozivanja rutina, u DE registar par treba postaviti adresu početka poruke umanjenu za jedan. Takode je potrebno da se sedmi bit bajta koji prethodi poruci i sadmi bit poslednjeg bajta poruke postava na jadan. Pogodnost koju pruža ova rutina je mogućnost ispisivanja željana poruke iz grupe poruka. Pri toma je potrebno u A registar postaviti redni broj poruka, a u DE registar adresu prve poruke umanjane za jedan. Kontrola pozicije ispisivanja i boje su dozvoljane u poruci. Karakteri i imena naredbi čiji su kodovi veći od 80h (sedmi bit postavljen na 1) ne mogu se ispisivati jer sa interpretiraju kao krajavi poruka.

Sledeći primer treba izvršiti kada je vrednost 0 unata u akumulator. To se ostvaruje u trećoj naredbi. Na taj način će se ispisati prva poruka. Punjenjem akumulatora vrednošću 1 i izvršavanjem programa **(POKE 30006,1:RANDOMIZE USR 30000)** ispisaće sa druga poruka na poziciji koja je odredena prethodnim ispisivanjem

edresa	kôd	mnemonike	komentar
7530	3E02	**LD A,02**	
7532	CD0116	**CALL 1601**	
7535	3E00	**LD A,00**	A = broj poruke
7537	113E75	**LD DE,753E**	
753A	CD0A0C	**CALL 0C0A**	ROM rutina PO--MSG
753D	C9	**RET**	
753E	80	**DEFB 80**	(128) Bit-7 = jedan
753F	16050A	**DEFB 16,05,0A**	
7542	1105	**DEFB 11,05**	
7544	4D696B72	**DEFB "M", "i", "k", "r"**	
7548	EF	**DEFB "o" + 80**	Kôd slova "o" + 128
7549	4B6E6A	**DEFB "K", "n", "j"**	
754C	696761	**DEFB "i", "g", "a"**	
754F	2038	**DEFB " ", "8"**	
7551	B5	**DEFB "5" + 80**	Kôd broja "5" + 128

RUTINE ISPISIVANJA BROJNIH VREDNOSTI. Ove rutine predstavljaju drugu grupu rutina za ispisivanje. Sledeća rutina ispisuje celobrojne vrednosti brojeva iz opsega od 0 do 9999. Preuzima se sadržaj BC registar para i ispisuje se se uslovima koji se mogu definisati kao i u prethodnim slučajevima pomoću kontrolnih kodova i RST 10 naredbe.

Sledeći primer ilustruje ispisivanje sedržaja BC registar para.

adrese	kôd	mnemonika	komentar
7530	3E02	**LD A,02**	
7532	CD0116	**CALL 1601**	
7535	01C107	**LD BC,07C1**	BC = 2000
7538	CD1B1A	**CALL 1A1B**	ROM rutina OUT--NUM-1
753B	C9	**RET**	

Rutina ispisivanja brojeva predstavljenih u petobajtnoj formi je rutina koja pokriva sve brojeve koje Spektrum moža ispisati. Rutina preuzima patobajtnu formu broja sa računarskog steka, tj. memorijskog prostora u koji je smeštana vrednost izračunata korišćenjem računarskog dela monitorskog programa. U sledećem primeru vidi se upotreba računarskog dela gde sa sadržaj registra A stavlja na računarski stak (četvrta naradba) uz pozivanje ROM rutine za izračunavanje kvadratnog korena sadržaja registra A (peta, šesta i sedma naredba). Dobijeni razultat, koji je smešten na steku, preuzima se od rutine ispisivanja svih brojnih vrednosti. Značajno je da se pri tome sadržaj steka gubi.

adresa	kôd	mnemonika	komentar
7530	3E02	**LD A,02**	
7532	CD0116	**CALL 1601**	
7535	3E03	**LD A,03**	
7537	CD282D	**CALL 2D28**	A na računarski stak
753A	EF	**RST 28**	ROM rutina FP--CALC
753B	28	**DEFB 28**	Kvadratni koren
753C	38	**DEFB 38**	Kraj računanja
753D	CDE32D	**CALL 2DE3**	ROM rutina PRINT--FP
7540	C9	**RET**	

5-3 RUTINE CRTANJA

Ova grupa rutine omogućuje crtanje u visokoj rezoluciji. To su rutine koje omogućuju određivanje stanja svaka od 256 × 176 tačaka slike na ekranu. Postoje rutina crtanja tačke, linije i kruga.

CRTANJE TAČKE se može ostvariti postavljanjem x i y koordinate, u koordinatnom sistemu koji važi i za bajzik **PLOT** naradbu, u BC registar par ili na računarski stek uz pozivanje odgovarajuće rutine. U prvom slučaju se u B ragistar postavlja vrednost y, a u C registar x koordinata. Poziv odgovarajuće rutina se obavlja preko adrese 22E5h (PLOT–SUB).

U drugom slučaju se na računarski stek stavlja prvo koordinata x, a zatim y koordinata. Pozivanje rutine iscrtavanja ostvaruje se preko adresa 22DCh (PLOT).

CRTANJE PRAVE LINIJE može se ostvariti postavljanjem x i y koordinata na računarski stek uz pozivanja na adresu 2477h. Pre povratka u bajzik neophodno je povratiti vrednost u alternativnom HL registru, koja odgovara adresi naredna labele računarskog dela. Sledači primer za crtanje prave linija između tačaka 10,20 i 240+10,96+20 ilustruja izloženo.

adresa	kôd	mnemonika	komentar
7530	FD36430A	**LD (IY+43),0A**	U sistemske promanljive CORDS
7534	FD364414	**LD (IY+44),14**	početne koordinata
7538	3EF0	**LD A,F0**	x koord posladnje tačke
753A	CD282D	**CALL 2D28**	na stek (STACK A)
753D	3E60	**LD A,60**	y koordinata poslednje tačke
753F	CD282D	**CALL 2D28**	na stek
7542	D9	**EXX**	Sačuvati HL'
7543	E5	**PUSH HL**	–
7544	D9	**EXX**	–
7545	CD7724	**CALL 2477**	ROM rutina LINE–DRAW
7548	D9	**EXX**	Vratiti HL'
7549	E1	**POP HL**	–
754A	D9	**EXX**	–
754B	C9	**RET**	

CRTANJE ZAKRIVLJENJE LINIJE obavlja se postavljanjem na računarski stek vrednosti x i y koordinata krajnje tačke i parametra g koji definiše ugao promene pravca u radijanima. Rutina se poziva prako adrese 2394h. Sladeći primer ilustruje crtanja luka sa početnom tačkom 10,20 uglom 1 radijan i krajnjom tačkom 250,116.

adrese	kôd	mnemonika	komentar
7530	FD36430A	**LD (IY+43),0A**	U sistemsku promenljivu CORDS
7534	FD364414	**LD (IY+44),14**	početne koordinete
7538	3EF0	**LD A,F0**	x koordinata poslednje tačke
753A	CD282D	**CALL 2D28**	na stek (STACK A)
753D	3E60	**LD A,60**	y koordinata poslednje tačke
753F	CD282D	**CALL 2D28**	na stek
7542	3E01	**LD A,01**	g=1
7544	CD282D	**CALL 2D28**	na stek
7547	D9	**EXX**	Sačuveti HL'
7548	E5	**PUSH HL**	–
7549	D9	**EXX**	–
754A	CD9423	**CALL 2394**	ROM rutina DR-3-PRMS
754D	D9	**EXX**	Vratiti HL'
754E	E1	**POP HL**	–
754F	D9	**EXX**	–
7550	C9	**RET**	

CRTANJE KRUGA iz mašinskog programa obavlja se postavljanjem koordinata x, y i poluprečnika r na računski stek. Pri tome nije potrebno postavljanje vrednosti koordineta poslednje tačke iz prethodnog iscrtavanja. Uz navedene izmene i izvršavanje rutine iscrtevanja kruga **(CALL 232Dh)** prethodni primer može poslužiti kao ilustracija.

U opštem slučeju iscrtavanje zakrivljene linije i kruga potrebne paremetre nije moguće uneti na računarski stek preko A registra. Tada perametre treba definisati preko A, E, D, C.i B registera uz poziv rutine na adresi 2AB6h (STACK STORE).

5-4 RUTINE ZVUKA

Upravljenje redom zvučnika u Spektrumu obavlje sema centrelna procesorska jedinica. Rutine koje se pri tome koriste nalaze

se u memoriji od lokacije Ø3B5h. Upotreba zvučnika može se ostvariti iz mašinskog programa na dva načina. U jednom se potrebne vrednosti za trajanje i visinu tona unose preko HL i DE registar parova, a u drugom preko vrednosti postavljenih na računarski stek.

Prvi način aktiviranje, preko adrese Ø3B5h zehteva u DE registar paru vrednost koja odgovare proizvodu željene frekvencije i trajanja tona. Trajanje tone je izraženo u sekundama. Vrednost sadržeja HL registar para se nalazi prema izrazu:

437500/f–30.125 f–frekvencija tona.

To je ilustrovano u sledećem primeru za ton frekvencije 1000 Hz i trejanja 1s.

adresa	kôd	mnemonika	komentar
7530	11E803	**LD DE, Ø3E8**	
7533	**2197Ø1**	**LD HL,Ø197**	
7536	**CDB5Ø3**	**CALL Ø3B5**	ROM rutine BEEPER
7539	**C9**	**RET**	

Drugi način aktiviranja zvučnika: Na računarski stek se postavljaju dve vrednosti, od kojih prva odgovara trajanju izraženom u sekundama, e druga visini tona. Oba parmaetra odgovaraju bejzik komandi **BEEP** trajanje, visina. Izvršavanje rutine se obavlja pozivanjem potprograma na adresi parametra **BEEP** Ø3F8h.

5-5 RUTINE SNIMANJA

Rutine snimanja, učitavanja i provere blokova su rutine sa mogućnošću direktnog pozivanja iz korisničkih mašinskih programa. U svim slučajevima dužina bloka, izražena u bajtime, odredena je sadržajem DE registar pera. Sadržaj IX register para ukazuje na adresu prvog bejte bloka. Sadržaj A registra ima vrednost ØØ za zaglavlje i 255 za blok bejzik ili mašinskog programa. Sledeća dva programa prikazuju sva tri slučeja.

Rutina snimenja bloka:

adrese	kôd	mnemonika	komenter
7530	3EFF	**LD A,FF**	**FFh za blok**
7532	**DD21ssss**	**LD IX, start**	**Početak bloka**
7536	11dddd	**LD DE,** dužina	Dužina bloka

7539	CDC204	**CALL #4C2**	ROM rutina SAVE-BYTES
753C	C9	**RET**	

Adresa	Rutina učitavanja bloka: kôd	Mnemonika	Komentar
7530	37	**SCF**	BIT pranosa jedinica za učitavanje
7531	3EFF	**LD A,FF**	FFh za blok
7533	DD21ssss	**LD IX**, start	Početak bloka
7537	11dddd	**LD DE**, dužina	Dužina bloka
753A	CD5605	**CALL #556**	ROM rutina LOAD–BYTES
753D	C9	**RET**	

Gore navadena rutina se primanjuje i za proveru bloka, sa razlikom što ja bit prenosa postavljan na nulu.

5-6 RUTINE DEKODOVANJA TASTATURE

Monitorski program obezbeđuje očitavanje tastatura na sledeći način: Pri pojavi signala prekida zaustavlja se izvršavanja tekućeg programa i prelazi sa na izvršavanje potprograma na adrasi 38h. Signal prekida sa javlja 50 puta u sekundi. Potprogram takođe povećava vradnost sadržaja sistemske promenljive FRAMES.

Za očitavanje tastatura koristi sa viša rutina iz potprograma na adresi 38h. Rezultat njihovog izvršavanja je kôd pritisnutog tastera koji sa smešta u sistemsku promenljivu LAST K. Peti bit sistemska promenljive na adresi 23611 postavlja se na jadan da bi sa znalo da je pritisnut novi taster. Izvršavanje ovog potprograma je onemogućeno naredbom DI u toku izvršavanja naredbi **LOAD, SAVE, VERIFY, MARGE** i **BEEP.** U slučaju prve četiri naredbe posebna rutina proverava da li je pritisnut **BREAK** tastar. **BEEP** naredba nije predviđena da bude prekidana.

Pri očitavanju tastatura treba razlikovati dva slučaja. Prvi sa odnosi na programe pri čijem izvršavanju je ostavljena mogućnost prekida, a u drugom slučaju se zahteva onemogućanje nastajanja prekida (**DI**). U prvom slučaju program koji očitava tastaturu može biti sledeći:

Adr.	**RES 5,(IY + 1)**	Priprema bita 5
Adr. + 4	**HALT**	Čekenje prekida
Adr. + 5	**BIT 5,(IY + 1)**	Taster pritisnut?
Adr. + 9	**JR Z,AP**	Ne. Odlazk ne predviđenu adresu
Adr. + 11	**LD A,(LASTK)**	Da. U A je kôd tastera
	-	-
	-	-

Umesto **HALT** naredbe kojom se čeka pojava signala prekida i odlaska ne rutinu sa adresom 38h, može se primeniti naredba **RST 38h**. Na tej način se postiže direktan prelazak na izvršavanje rutine.

U drugom slučaju, kada je potrebno onemogućiti prekid, mašinske rutina za očitavanje testature može biti sledeća:

Adr.	**RES 5,(IY + 1)**	Priprema bita 5
Adr. + 4	**RST 38**	Očitevenje tastature
Adr. + 5	**DI**	Onemogućavanje prekida
Adr. + 6	**BIT 5,(IY + 1)**	Taster pritisnut?
Adr. + 10	**JR Z,AP**	Ne. Odlazak na predviđenu lokaciju AP
Adr. + 12	**LD A,(LASTK)**	Da. U A je kôd tastera
	-	-

Naredba **BIT 5,(IY + 1)** testira vrednost petog bita sistemske promenljive na adresi 5C3Bh. S obzirom da je vrednost sedržaja IY registra 5C3Ah (23610) adresa promenljive FLAGS (23611) i IY + 1. Naredbom **DI** u drugom slučaju poništava se dejstvo **EI** naredbe kojom se završava **RST 38h**, rutina i obezbeduje se dalja zebrana prekida.

Na osnovu svega što je do sada izloženo, čitaocu neće predstavljati teškoću da nepravi demonstracione progreme za očitavanje testature upotrebom ROM rutine **RST 38h**, a takođe i de je upotrebi u svojim programima. Ranije navedeni nečini direktnog očitavanja tasteture upotrebom ulaznih **(IN)** naredbi pogodniji su za korišćenje pri testiranju manjeg broja tastera.

5-7 RUTINE ZA RAČUNANJE

U delu ROM-a od lokecije ?F9Bh do lokacije 386Dh nalaze se rutine koje obezbeđuju postojeće Spektrumove matematičke naredbe i naredbe sa stringovima Taj deo se naziva i kalkulator (eng

floting-point calculator). Sastoji se iz rutina za 66 operacija se pokretnim zarezom. Pri tome se koriste tri tipe operacije:
- binarne (nad dva argumenta; npr. sabiranje)
- unarne (nad jednim argumentom; npr. trigonometrijske funkcije)
- manipulacione (vrši se kopiranje argumeneta).

Operacije se izvršavaju nad argumentima koji se nalaze u delu RAM memorije koji se naziva računerski stek (eng. calculator stack). Njegov početak je određen sistemskom promenljivom STKBOT, a kraj sa promenljivom STKEND. Satavljanjem jednog, odnosno dva argumenta na stek i izvršavljnjem unarne, odnosno binarne operacije na steku osteje samo rezultat izvršene operacije.

Za stavljanje vrednosti argumenata na stek koriste se rutine na sledećim adresema.

- 2D28h (STACK A); vrednost u A registru se prevodi u petobajtnu formu sa pokretnim zarezom i stavlja na stek
- 2D2Bh (STACK BC); vrednost u BC registar paru, u apsolutnoj binarnoj formi, prevodi se u petobajtnu formu sa pokretnim zarezom i stavlja na stek
- 2AB6h (STK-STORE); vrednosti u A, E, D, C i B registrima stavljaju se na stek u petobajtnoj formi, gde prva vrednost (A reg.) određuje eksponent.

String artumenti se takođe petobajtno predstavljaju. Prvi bajt se ne more koristiti (reg. A u slučeju STK-STORE rutine). Sledeća dva određuju adresu početka stringa, a dva poslednje njegovu dužinu.

Za vraćanje izračunatih vrednosti sa računarskog steka u registre koriste se rutine na sledećim adresama:

- 2DD5h (FP TO A); poslednja stavljena ili izračunata vrednost sa steka zaokružuje se na najbliži ceo broj i stavlja u A registar; bit prenosa se postavlja na jedinicu eko dolezi do premešenje, a bit nule na nulu za negativnu vrednoet
- 2DA2h (FP TO BC); BC registar par se puni poslednjom vrednošću sa steka; bit prenosa se postavlja na jedinicu za premešenje, a bit nule na nulu za negativnu vrednost
- 2BF1h(FP TO AEDCB); Registri A, E, D, C i B se pune vrednostima bajta poslednje vrednosti na steku, bilo da je u pitanju numerička ili string promenljiva.

Pozivanje rutine za izvršavanje pojedinih operacija može se ostvariti uz pomoć jednobajtnih vrednosti (literali) koje označavaju

svaku od 66 operacija. Prva naradba mora biti RST 0028h, koja poziva rutine računanja, za kojom sledi niz literala odgovarajućih operacija. Poslednji literal u nizu mora biti 38h da označi kraj pozivanja rutina za računanje.

U sledačoj tabeli su date numaričke vrednosti-literali odgovarajućih operacija i oparacije većeg dela postojećih rutina računanja.

Litaral	Operacija
00	Skok ako je rezultat prethodne operacija istinit. Naredna numerička vrednost definiše sladaći literal (ekvivalentno naredbi **JR**).
01	x y, zamena mesta poslednje dve vrednosti na steku
02	Brisanje poslednje vrednosti na steku
03	x−y; oduzimanje poslednje od pretposlednje vrednosti na steku
04	x*y; množenje pretposlednje i poslednje vrednosti
05	x/y; deljenje pretposlednje i poslednje vrednosti
06	x↑y, stepenovanje poslednje vrednost na pretposlednju
07	xORy, logičko sabiranje
08	xANDy, logičko množenje
09	x⩽y, poređenja pretposlednje i poslednje vrednosti
0A	x⩾y,
0B	x≠y,
0C	x>y;
0D	x<y;
0E	x=y;
0F	x+y, sabiranje pretposlednje i poslednje vrednosti
10	x$ANDy, logičko množenje stringa (pretposlednje) i brojne (poslednje) vrednosti
11	x$⩽y$, poređenja stringova određanih pretposlednjom i poslednjom vrednošću na steku
12	x$⩾y$,
13	x$≠y$,
14	x$>y$,
15	x$<y$,
16	x$=y$,
17	x$+y$, sabiranje stringova
18	USR y$; y$ je između a(A) i u(U)
1A	y$=INKEY$
1B	y=−y

1C	y=CODE y$(1 TO 1); na stek se stavlja kod prvog karaktera
1E	y=LEN y$
1F	y=SIN y
20	y=COS y
21	y=TAN y
22	y=ASN y
23	y=ACS y
24	y=ATN y
25	y=LN y
26	y=EXP y
27	y=INT y
28	y=SQR y
29	y=SGN y
2A	y=ABS y
2B	y=PEEK y
2C	y=IN y
2E	y$=STR$ y
2F	y$=CHR$ y
30	y=NOT y
31	Manipulaciona operacija koja pravi kopiju poslednje vrednosti (x=y).
32	xMODy; x=INT(x/y); y=x-INT(x/y)
33	Bezuslovni skok, ostalo isto kao kod litarala 00.
34	STK-DATA; na stek se u petobajtnom obliku postavlja vrednost data, u kompresovanoj formi, narednim literalima
36	y<0
37	y>0
38	Prestanak interpretiranja narednih kodova kao literala.
3D	Poslednja vrednost na steku pretvara iz celobrojne petobajtne forme u petobajtnu formu sa pomičnim zarezom.
A0	0
	Stavljanje navedenih vrednosti na vrh steka
A1	1
A2	1/2
A3	PI/2
A4	10

U navedenim operacijama upotrebljene su sledeće oznake:

- x – Pretposlednja brojna vrednost postavljena na stek
- y – Poslednja brojna vrednost postavljena na stek
- x$ – Pretposlednja vrednost koja predstavlja string, stavljena na stek
- y$ – Poslednja vrednost koja predstavlja string, stavljena na stek.

Sledeći primer demonstrira upotrebu rutina računanja za izračunavanje vrednosti izraza $x^2/\sin x$ (ne zaboraviti otvaranje željenog kanala za ispisivanje).

```
LD      A,x     u akumulator staviti vrednost x
CALL    2D28    prebacivanje na stek
RST     28      upotreba ROM rutina za računanje
DEFB    31      dupliranje vrednosti  (x,x)
DEFB    31      isto                  (x,x,x)
DEFB    04      množenje              (x*x,x)
DEFB    01      zamena mesta          (x,x*x)
DEFB    1F      sin x                 (sin x,x*x)
DEFB    05      deljenje              (x*x/sin x)
DEFB    38      kraj
CALL    2DE3    ispisivanje
RET
```

Dati kratak prikaz određenih rutina za računanje ima za cilj upoznavanje sa jednim obimnim delom ROM-a. Detaljni prikaz bi iziskivao veliki prostor što postavlja pitanje opravdanosti s obzirom na postojeću neophodnu literaturu za potpuno razumevanje rutina računanja (3).

5–8 KANALI I STRIMOVI

Na ovom mestu će se detaljnije razmotriti deo operativnog sistema namenjenog za razmenu podataka između računara i perifernih uređaja.

Strim (eng. stream) predstavlja tok podataka između računara i perifernih uređaja (tastatura, štampač, ekran...). Kanali (engl. channel), odnosno kanalski programi, omogućavaju razmenu podataka sa tim perifernim jedinicama.

Strimovi su označeni brojevima od FDh do 10h. Operativni sistem ih raspoznaje tako što iz područja sistemskih promenljivih, označenog sa STRMS, koje počinje od adrese 23568, dužine 38 bajta, uzme relativnu adresu kanala koji je pridružen traženom strimu u području podataka o kanalima. Inicijalno su instalirani strimovi od FDh do03h. Ostalih 12 strimova korisnik može po potrebi sam da instalira. Područje STRMS inicijalno izgleda ovako:

5C10	0100	strim FDh vodi ka kanalu 'K' (tastatura)
5C12	0600	strim FEh vodi ka kanalu 'S' (ekran)
5C14	0B00	strim FFh vodi ka kanalu 'R' (radni prostor)
5C16	0100	strim 00h vodi ka kanalu 'K'
5C18	0100	strim 01h vodi ka kanalu 'K'
5C1A	0600	strim 02h vodi ka kanalu 'S'
5C1C	1000	strim 03h vodi ka kanalu 'P' (štampač)
5C1E	xxxx	nedefinisano
		slobodno područje
5C36	xxxx	nedefinisano

Adresa ulaza za neki od strimova izračunava sa po formuli:
5C00h + (broj strima *2+18h) mod FFh=adrasa ulaza u STRMS

Kanali su opisani područjem podataka o kanalima. Po uključenju Spektruma promenljiva CHANS sadrži adrasu 5CB5h od koje počinju informacije o kanalima koje ovako izgledaju:

5CB6	F409	PRINT-OUT izlazna rutina kanala 'K', na adresi 09F4h
5CB8	A810	KEY-INPUT ulazna rutina kanala 'K', ne adrasi 10A8h
5CBA	4B	ASCII kôd slova 'K', ime kanala K
5CBB	F409	PRINT-OUT izlazna rutina kanala 'S' ne adresi 09F4h
5CBD	C415	REPORT-J ulazna rutina kanala 'S', na adrasi 15C4h
5CBF	53	ASCII kôd slova 'S', ime kanala S
5CC0	810F	ADD-CHAR izlazna rutina kanala 'R' na adresi 0F81h
5CC2	C415	REPORT-J ulazna rutina kanala 'R', na adresi 15C4h

5CC4	52	ASCII kôd slova 'R', ime kanala R
5CC5	F4Ø9	PRINT-OUT izlazna rutina kanala 'P', na adresi Ø9F4h
5CC7	C415	REPORT-J ulazna rutina kanala 'P', na adresi 15C4h
5CC9	5Ø	ASCII kôd slova 'P', ime kanala P
5CCA	8Ø	end marker,

Rutina PRINT-OUT je kanalski program koji opslužuje više različitih periferija u zavisnosti od postavljenih pokazatelja (flegova). Ona obavlja sva ispisivanja na gornji deo ekrana, u editorsko područje ekrana (donji deo) i na Spektrumov štampač.

Rutina KEY-INPUT je kanalski program koji opslužuje tastaturu i vraća kôd poslednjeg pritisnutog tastera, a pri tome ne vraća kôd CAPS-LOCK tastera niti pak kodove za promenu načina rada ili boje, već te funkcije obavlja sam kanalski program.

Rutina ADD-CHAR je kanalski program koji ispisuje karakter u radno područje ekrana prilikom editovanja ili INPUT naredbe.

U slučaju kada neki kanal nema neki od smerova prenosa podataka, rutina za taj smer je REPORT-J. Ona javlja grešku ako pokušamo da prenesemo podatke u tom smeru.

Za instaliranje novog kanala potrebno je urąditi sledeće:

a) Sistemske promenljive VARS, PROG, NXTLIN, DATADD i ELINE uvećati za broj novih kanala puta tri.
b) Dodati podatke za nove kanale, po napred navedenom rasporedu, počev od pozicije end marker, a njega staviti iza poslednjeg bajta podataka o kanalima.
c) Odgovarajuće mesto u memoriji popuniti kanalskim programima.
d) Instalirati nove strimove prema gore datom.

6 Hardver

Spaktrumove elektronske kola možemo podeliti u 4 velike celina: mikroprocesor, memoriju, kola za vezu sa perifernim jedinicama i nepajanje. Memoriju čine ROM memorija i dve celine RAM memorije: 16K RAM i 32K RAM (32K RAM postoji samo u Spektrumu sa 48K RAM-a). Veliki broj kola ze vezu se perifernim jedinicama je integrisan u jedno kolo-ULA. Preko tog kole mikroprocesor ima vezu sa testaturom, TV akranom, zvučnikom i kasetofonom.

Na slici 6-1 su prikazeni glavni dalovi Spektruma i veze između njih: magistrala podataka, edresna magistrale i razne kontrolne linije.

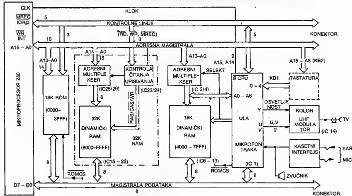
U nastavku će biti opisan sveki deo Spektruma pojedinačno.

6-1 MIKROPROCESOR

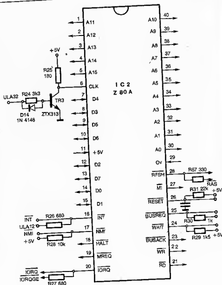
Spektrumova centralne procasorske jedinica je osmobitni mikroprocesor Z80A. Razvijen je u američkoj firmi Zilog proširivanjem mikroprocesora 8080 firme Intel, tako da je broj naredbi koje mogu da budu izvršene povečan na 158.

Sve oparecije koje mikroprocesor obavija su vremenski sinhronizovana u odnosu na klok (tekt). To je signal pravougaonog oblika i stabilne frekvencija koja je kontrolisana kristelom kvarca. U Spektrumu, Z80A redi na 3,5 MHz. Postoje i druga varijante ovog mikroprocesora: prethodnik Z80 koji može da radi ne učestanostima do 2,5 MHz i noviji Z80B kome je gornja granica 6 MHz.

Klok se generiše u ULA i šelja mikroprocesoru, ali kod tog signala promena se logičke nule na logičku jedinicu i obrnuto nisu dovoljno brze (strmina prelaza ja nedovoljna). Kako su svi događaji sinhronizovani sa trenutkom prelaze, oni moraju da budu što precizinije odredani, što strmiji. TR3 ime ulogu da te prelaske učini dovoljno brzim.



Slika 6–1 Blok šema Spektruma



Slika 6–2 Mikroprocesor

FUNKCIJE POJEDINIH IZVODA PROCESORA

Izvodi mogu da budu ulazi ili izlazi ili i jedno i drugo. Pojedini izlazi imaju mogućnost da predu u stanje visoke impedanse, čime omogućavaju nekoj drugoj jedinici da upravlja stanjem na liniji (engl. tristate-tristejt izlazi). Iznad simbola pojedinih izvoda se nalazi crta koja označava da je za njega aktivno stanje (ono pri kome vrši svoju funkciju) logička nula, a normalno, neaktivno stanje je logička jedinica.

–D0 do D7-magistrala podataka, dvosmerne linije (ulazi ili izlazi), imaju mogućnost da budu u stanju visoke impedanse. Preko ovih osam izvoda se razmenjuju podaci između mikroprocesora i memorije ili ulazno-izlazne jedinice.

–A0 do A15-adresna magistrala, izlazi, imaju mogućnost da budu u stanju visoke impedanse.
Preko ovih 16 izvoda mikroprocesor može da adresira 65 536 (2^{16}) lokacija u memoriji. Kada direktno komunicire sa ulazno-izlaznim jedinicama koristi samo A0 do A7 tako da postoji 256 različitih ulazno-izlaznih adresa.

–$\overline{\text{MREQ}}$ (engl. memory request-zahtev za memorijom), izlaz, može da se nalazi u stanju visoke impedanse.
Logička nula na njemu označava de se na adresnoj magistrali nalazi adresa lokacije u memoriji u koju se upisuje ili se iz nje čita podatak.

–$\overline{\text{IORQ}}$ (engl. input-output request-zahtev za ulazno-izlaznom jedinicom), izlaz, može da se nalazi u stanju visoke impedanse.
U trenutku kada se na njemu nalazi logička nula, na adresnim linijama se sigurno nalazi adresa ulazno-izlazne jedinice sa kojom se izvodi operacija upisivanja ili čitanja. Prema tome, $\overline{\text{MREQ}}$ je aktivan kada mikroprocesor komunicira sa memorijom, a $\overline{\text{IORQ}}$ kada komunicira sa ulazno-izlaznim jedinicama.

–$\overline{\text{RD}}$ (engl. read-čitanje), izlaz, mogućnost stanja visoke impedanse.

Aktivno stanje (logička nula) na ovom izvodu označava da centralna procesorska jedinica iščitava podatak iz memorije ili iz ulazno-izlazne jedinice.

–$\overline{\text{WR}}$ (engl. write-upiši), izlaz, mogućnost stanja visoke impedanse.

Aktivno stanje (logička nula) na ovom izvodu označava da centralna procesorska jedinica upisuje podatak u memoriju ili u ulazno-izlaznu jedinicu.

–$\overline{\text{M1}}$ mašinski ciklus jedan, izlaz

Logička nula na njemu se javlja kada mikroprocesor učitava iz memorije kod naredbe.
Istovremenim aktiviranjem $\overline{M1}$ i $\overline{IORQ}$ sa označava odobravanje zahteva za prakid (interrupt acknowledge cycle).
$-\overline{RFSH}$ (engl. refresh-osvežavanje), izlaz
Dinamičke RAM memorija, kakve su upotrebljene u Spektrumu, moraju da budu osvežavane najmanje jednom svake 2ms, da se njihov sadržaj ne bi izbrisao. Ovaj izvod svojim aktivnim stanjem (logička nula) označava da se na nižih sadam bita adresna magistrala (A0-A6) nalazi adresa potrebna za osvežavanje memorije.
$-\overline{HALT}$, izlaz
Svojim aktivnim stanjem (logička nula) označava da je izvršena programska naredba HALT. Posle ovoga centralna procasorska jedinica se zaustavlja i čeka zahtev za prekid (interrupt) od neke druge jedinica, i tek tada nastavlja sa radom.
$-\overline{WAIT}$, ulaz
Spore memorija i ulazno-izlazne jedinice logičkom nulom na ovom ulazu signaliziraju mikroprocesoru da sačeka izvesno vreme. Kada sa na njemu ponovo pojavi logička jedinica, mikroprocesor nastavlja sa radom.
$-\overline{RESET}$, ulaz
Ovaj signal (aktivno stanje je logička nula) postavlja centralnu procasorsku jadinicu u početno stanje. U Spektrumu taj signal se dobija u trenutku uključivanja napajanja pomoću kondenzatora C27 i otpornika R31. U početnom trenutku na kondenzatoru je nizak napon koji resetuje mikroprocesor. Taj napon raste, jer se kondenzator puni kroz otpornik, i kada preda odredenu vrednost mikroprocesor počinje da radi.
$-\overline{BUSREQ}$ (engl bus request-zahtev za magistralom), ulaz. Preko ovog ulaza neka druga jedinica signalizira (logičkom nulom) da želi da preuzme kontrolu nad sistemom. Kada primi ovaj signal, mikroprocesor po završetku tekućeg mašinskog ciklusa sve izlaze, koji to mogu, postavlja u stanja visoka impedanse i preko izlaza $\overline{BUSACK}$ potvrđuje jedinici da može da preuzme kontrolu.
$-\overline{BUSACK}$ (engl. bus acknowledge-potvrda zahteva za magistralom), izlaz.
Logičkom nulom na ovom izvodu centralna procesorska jedinica potvrđuje jedinici koja je uputila zahtev da može da preuzme kontrolu nad sistemom.

-$\overline{INT}$ (engl. interrupt-prekid), ulaz, zahtev za prekidom
Kada se na ovom ulazu pojavi logička nula, mikroprocesor prekida sa izvršavanjem tekućeg programa i prelazi na izvršavanje drugog koji je predviđen za ovu svrhu. Na ovaj način se omogućava da mikroprocesor najbrže moguće reaguje na neke spoljašnje signale, što je u nekim primenama vrlo značajno. U samom mikroprocesoru se nalazi jedna memorijska ćelija (flip-flop) od čijeg stanja (Ø ili 1) zavisi da li će prekid biti omogućen ili ne. Time se mikroprocesor štiti od prekida kada on nije poželjan. Istovremeno aktiviranjem $\overline{M1}$ i $\overline{IORQ}$ označava da se izvršava ciklus potvrde prekida (interrupt acknowledge cycle).
-$\overline{NMI}$ (engl. non-maskabla interrupt-nemaskirajući prekid), ulaz.
Ovo je ulaz koji aktivira negativna ivica (prelazak sa logičke jedinice na logičku nulu). Ovaj zahtev za prekidom se uvek prihvata tj. ne može se maskirati kao prethodni. Tada po završetku tekuće naredbe mikroprocesor počinje da izvršava program koji počinje na adresi ØØ66h

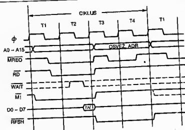
VREMENSKI DIJAGRAMI

Pomoću vremenskih dijagrama se prikazuju signali (naponi) na pojedinim izvodima mikroprocesora za vrame obavljanja pojedinih operacija. Rafarantni signal za sve događaje je klok (takt), jer se sve dešava sinhrono sa njim. Jedan period kloka se naziva T ciklus.

Mikroprocesor izvršava program tako što iz memorija učita kôd naredbe, izvršava tu naredbu, učitava kôd sladeća naradbe, izvršava je i tako redom. Ceo ovaj period učitavanja i izvršavanja jedne naredbe traje jedan ili više tzv. mašinskih ciklusa. Jedan mašinski ciklus traje nekoliko T ciklusa. Kada su ostali dalovi računarskog sistema suviše spori, mogu preko $\overline{WAIT}$ ulaza da zatraža da sa između drugog i trećag T ciklusa ubaci jedan ili više novih T ciklusa čiji je zadatak da produže mašinski ciklus.

Prvi mašinski ciklus uvek je M1 ciklus koji se sastoji od četiri do šest T stanja (sem ako nije produžan aktiviranjem $\overline{WAIT}$ ulaza) u kome mikroprocesor učitava kôd naradbe.

Na slici su prikazani vremanski dijagrami nakoliko signala za vreme M1 ciklusa.
Ovaj ciklus se sastoji od četiri T ciklusa: T1, T2, T3 i T4. U ciklusu T1 počinju pripreme za učitavanja naredbe. Signal M1 prelazi u aktivno stanja (logička nula) da bi naznačio da se učitava kôd naredbe, a na



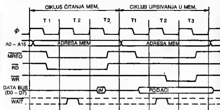
Slika 6–3 Vremenski dijagram ciklusa M1

adresnu magistreiu se postavija adresa memorijske lokacije u kojoj
se nalazi žeijene neredbe. Zatim mikroprocesor čeka poia T cikluse
da bi dozvolio de se adresa stabilizuje, pa aktivira siedeće signale:
$\overline{MREQ}$, čime označeva da se podatak razmenjuje sa memorijom i
$\overline{RD}$, da bi odredio da se radi o procesu učitavanja. Ako se u ciklusu
T2 signal $\overline{WAIT}$ nalazı na nivou iogičke nule, izmedu T2 i T3 će biti
ubačeno još nekoliko T ciklusa. U ovom siučeju je negiašeno da se
$\overline{WAIT}$ naiazi u neaktivnom stanju (logičke jedinice), dok njegovo sta-
nje u osteiim periodime ne utiče na rad, pa je označen isprekidanom
linijom. Do početka ciklusa T3 podaci iz memorije se stabilizuju na
magistraii podataka (D0-D7) tako da se u toku prednje ivice T3 oni
učitavaju u mikroprocesor. Odmeh zatim počinje period osveževa-
nje dinemičke RAM memorije. $\overline{M1}$, $\overline{MREQ}$ i $\overline{RD}$ preiaze u neektivno
stanje (iogička jedinica), mikroprocesor na adresnu magistralu po-
stavija edresu potrebnu ze osvežavanje i aktivira signai $\overline{RFSH}$, kojim
označava da je ciklus osvežavanja memorije u toku. Posle poie T
ciklusa, kada se signel na adresnoj magistraii stabilizovao, ektivira
se $\overline{MREQ}$ i time zepočinje osvežavanje. Ne sredini perioda T4 se
$\overline{MREQ}$ deaktivira, a na kraju T4 se deaktiviraju i adresne magistrale i
$\overline{RFSH}$. Time se završava mašinski ciklus M1 i započinje siedeći.
U toku perioda T3 i T4 u mikroprocesoru se vrši i dekodavanje učita-
ne naredbe i započinje njeno izvršavanje. Neke naredbe su tekve da

je ovaj period dovoljan i za njihovo izvršavanje, dok je za izvršavanja drugih potreban još jedan ili više mašinskih ciklusa

ČITANJE IZ MEMORIJE I UPISIVANJE U MEMORIJU

Na slici 6-4 su prikazani mašinski ciklusi čitanja iz mamorije i upisivenja u memoriju. Oba su dugačka po tri T ciklusa jer je u periodima T2 $\overline{WAIT}$ signal neaktivan (logička jadinica). U suprotnom bi između T2 i T3 bili ubacivani novi T ciklusi i procesor bi čakao sve dok sa $\overline{WAIT}$ signal ne deaktivira.



Slika 6–4 Vremenski dijagram čitanja iz memorije i upisivanja u memoriju

Čitanje iz memorije počinje postavljanjam adrese mamorijske lokacija iz koje se podaci čitaju na adrasnu magistralu. Zatim mikroprocesor čeka pola T ciklusa da bi sa naponi stabilizovali, pa aktivira signale $\overline{MREQ}$ kojim označava da komunicira aa mamorijom i $\overline{RD}$ koji je komande za čitanje podataka. Do sradina ciklusa T3 podaci su postali stabilni na magistrali podataka i tada ih mikroprocesor učitava. Odmah se daaktivireju signali $\overline{MREQ}$ i $\overline{RD}$ (vračaju na logičku jedinicu), a po završatku poslednjeg ciklusa T3 adresa sa uklanja sa adresne magistrale i postavlja nova.

Upisivanja u memoriju otpočinje takode postavljanjem adresa na adresnu magistralu u samom početku perioda T1. Zatim postoji pauza od pola T ciklusa da b[illegible] postavljani podaci stabilizovali. Na polovini ciklusa T1 se aktivira s[illegible]al $\overline{MREQ}$ (pralazi na logičku nulu)

čime se označava da se komunicira sa memorijom. Istovremeno mikroprocesor na magistralu podataka (DØ-D7) postavlja podatke koji treba da budu upisani u memoriju, a zatim čeka jedan T ciklus da bi se ti naponi stabilizovali. Na sredini ciklusa T2 se aktivira komanda upisa $\overline{WR}$ (prelazi na logičku nulu), a deaktivira se na sredini perioda T3 dovoljno pre uklanjanja podataka, tako da su za vreme trajanja $\overline{WR}$ oni sigurno stabilni. Tim signalom se podaci upisuju u memoriju. Sredinom perioda T3 se deaktivira i $\overline{MREQ}$, a na kraju T3 i adresa na AØ-A15 i podaci na DØ-D7 i time je proces upisivanja u memoriju završen.

ČITANJE ILI UPISIVANJE U PERIFERNU JEDINICU

Na slici 6-5 su prikazani vremenski dijagrami signala pri čitanju iz ulazno-izlazne jedinice i pri upisivanju u nju.

Slika 6-5 Vremenski dijagram čitanja iz periferne jedinice i upisivanja u periferenu jedinicu

Taj memorijski ciklus traje četiri T perioda. Jedan period čekanja T_w se automatski ubacuje, jer su ulazno-izlazne jedinice relativno spore. U slučaju da je u sredini perioda T_w aktiviran zahtev za čekanje $\overline{WAIT}$, bilo bi ubačeno još T_w perioda sve dok se taj signal ne bi deaktivirao. Na slici to nije slučaj, jer je $\overline{WAIT}$ neaktivan. Da bi označio da se komunikacija vrši sa ulazno-izlaznom jedinicom, a ne sa me-

morijom, mikroprocesor aktivira signal $\overline{IORQ}$ na početku perioda T2 a deaktivira ga na sredini T3. Adresa jedinice sa kojom se komunicira nalazi se na adresnoj magistrali u toku celog trajanja ovog mašinskog ciklusa. Pri čitanju podataka aktivira se signal $\overline{RD}$ početkom perioda T2, a deaktivira sredinom perioda T3, dok signal $\overline{WR}$ ostaje neaktivan za vreme celog ciklusa. Podatke mikroprocesor učitava sredinom perioda T3 jer su do tada postali stabilni na magistrali podataka.

Za vrame upisivanja u ulazno-izlaznu jedinicu mikroprocesor postavlja podatke na magistralu podataka sredinom perioda T1, a deaktivira ih tek na kraju mašinskog ciklusa, tako da su sigurno stabilni za vrame komande upisa $\overline{WR}$, koja se aktivira početkom perioda T2, a deaktivira sredinom perioda T3. Kao i prilikom čitanja, signal $\overline{IORQ}$ se aktivira početkom perioda T2, a deaktivira sredinom T3 da bi se naznačila komunikacija sa ulazno-izlaznom jedinicom.

6-2 RAM

RAM je skraćenica od engleskih reči random access memory, što znači da se radi o memoriji kod koje direktno možemo prići (razmenjivati podatke) bilo kojoj memorijskoj ćeliji. U nju možemo upisati podatak, ili ga iz nje pročitati.

RAM memorije su veoma brze, što znači da se podaci sa njima razmenjuju za veoma kratko vreme (nekoliko stotina ns). Kada nastane napajanje, sadržaj memorije se gubi.

Postoje dve vrste RAM-a: statički i dinamički. Memorijske ćelije statičkog RAM-a se sastoje od nekoliko tranzistora. Oni čine kolo koje ima dva stabilna stanja, jedno odgovara logičkoj jedinici, a drugo logičkoj nuli. Upisivanjem podataka kolo se prebacuje u odgovarajuće stanje i u njemu ostaje do upisivanja sledećeg podatka ili gubitka napajanja. Memorijska ćelija dinamičkog RAM-a se sastoji od jednog MOS (metal oxide semiconductor) tranzistora koji podatke pamti samo vrlo kratko vreme, tako da mora da se stalno osvežava (na primer svake milisekunde). Pri čitanju se podaci u ćeliji gube, pa moraju da odmah budu ponovo upisani. Zbog toga ovaj proces istovremeno ima i ulogu osvažavanja. Dinamički RAM je komplikovaniji za upotrebu, ali je pri većim kapacitetima jeftiniji od statičkog, jer se njegova memorijska ćelija sastoji od samo jednog tranzistora.

U Spektrumu se koristi dinamički RAM. Postoje dve verzije Spektruma: sa 16K (kilobaita) RAM-a i sa 48K RAM-a. RAM druge

verzije je organizovan u dve caline, 16K potpuno isti kao u prvoj verziji i dodatnih 32K.

16K RAM

Ova grupa se sastoji od deset integrisanih kola, osam memorijskih i dva multipleksara. Svako od memorijskih kola ima kapacitet od 16 kilobita (16384 × 1) i kod svakog od njih je linija za podatke vezana za po jednu liniju magistrale podataka tako da zajedno čine memoriju od 16 kilobajta (16384 × 8). Memorijske ćelije u njima su organizovane u 128 radova i 128 kolona. Broj izvoda na ovom integrisanom kolu smanjen je multipleksiranjem adrese, što znači da se prvo na njega dovodi jedan deo adrese, a za njim drugi. U ovom slučaju sedam adresnih linija kola vezano je za izlaze multipleksera (74LS157), čiji su ulazi vazani na adresnu magistralu. Signali $\overline{RAS}$ (row address stroba) najava adrese rada i $\overline{CAS}$ (coulmn address strobe) najava adrase kolone, koje generiše ULA, upravljaju ubacivanjem potrebne adrese u memoriju. Kada se pojavi $\overline{RAS}$, multiplekser prenosi adresu reda sa linija A0-A6 adresna magistrale na memorijska kola, a kada se pojavi $\overline{CAS}$, do kola stiže adresa kolone sa linija A7-A13. Adresne linije memorijskih kola su vezane za izlaza multipleksera preko otpornika od 330 oma, a za adresne linije koja dolaze iz ULA-a (DRAM A0-DRAM A6) direktno. Na ovaj način je obezbedeno da ULA ima prednost nad mikroprocesorom, što je potrebno zbog osvežavanja slike na TV ekranu. U jednom delu ove memorija se nalaze podaci o tome šta se na njemu prikazuje. ULA čita te podatke i na osnovu njih stvara signala koje šalje preko modulatora na televizor. Ukoliko centralna procesorska jedinica pokuša da istovremeno pristupi tom delu memorije, ULA će na osnovu stanja na linijama uvideti da dolazi do konfliktne situacije, pa će prestati da šalje klok. Na taj način će se stvoriti potrebna pauza u radu mikroprocesora, koji će odmah, čim ULA završi sa osvežavanjem slike, moći da komunicira sa tim dalom memorije.

Ova memorija gubi podatke ako se svakom redu ne pristupi barem jednom svake 2ms. Dok ULA osvežava sliku i pri tome iz nje uzima podatke, oni se istovremeno osvežavaju pa je sve u redu. Jedino u periodu stvaranja impulsa za vertikalnu sinhronizaciju slike postoji suviše velika pauza od 5 ms, pa u tom periodu memoriju osvežava centralna procesorska jedinica uz pomoć signala $\overline{RFSH}$ koji je preko otpornika vezan za $\overline{RAS}$

32K RAM

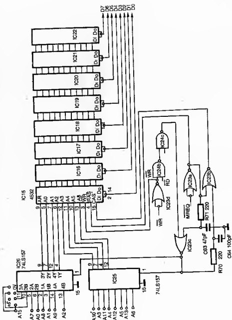
Verzija Spektruma sa 48K RAM-a ima 12 integrisanih kola više od 16K verzije, 8 memorijskih i 4 logička kola. Organizacija memorijskih kola je 32768 × 1 (kapacitet 32 kilobita). Ulazna i izlazna linija podataka svakog kola su spojene i vezane za jednu liniju magistrale podataka. Sva kola zajedno čine celinu od 32K bajta (32K × 8) čije su adrese od 32768 do 65535. Adresa je radi smanjenja broja izvoda i u ovom slučaju multipleksirana. Kada im je signal S (select) na logičkoj jedinici, dva multipleksera (74LS157) dovode na adresne linije kola adrese od A0 do A7, a kada je na logičkoj nuli, adresa od A8 do A14. Kontrolne signala $\overline{RAS}$ i $\overline{CAS}$ u ovom slučaju na generiše ULA nego nekoliko logičkih kola. Ona su smaštena u dva integrisana kola, u 74LS00 sa nalaze 4 NI kola (iskorišćena su dva), a u 74LS32 se nalaze 4 ILI kola. Adresna linija A15 je naaktivna (logička nula) kada ovaj dao memorijskog prostora nije adresiran tj. kada je adrasa od 0 do 32767. Zato je vezana za NI kolo IC24c tako de sprečava stvaranje drugih signala kada ja ona na logičkoj nuli.

Signal $\overline{RAS}$ memorijskih kola je $\overline{MREQ}$ malo zakašnjen kolom IC23a. Signal S multipleksera se dobija unošenjem kašnjenja u $\overline{RAS}$ pomoću otpornika R71 i kondenzator C63, a njegovo stvaranje će biti onemogućano pomoću IC23c ako nije aktiviren $\overline{WR}$ ili $\overline{RD}$. Signal $\overline{CAS}$ će se aktivirati signalom S koji je zakašnjen pomoću otpornika R70 i kondenzatora C64, a deaktivirati kada i $\overline{RAS}$ preko ILI kola IC23b.

Mamorljska kola će upamtiti adresu A0-A7 kada se aktivira $\overline{RAS}$ (preda na logičku nulu), zatim ća S preći na logičku nulu i dovesti na kola adrese A8-A14 koja ća biti zapamćene aktiviranjam signala $\overline{CAS}$.

Ako se uz $\overline{RAS}$ i $\overline{CAS}$ aktivira i $\overline{WR}$, memorijska kola će podatke sa magistrale podataka zapamtiti u adresiranoj lokaciji, a ako su $\overline{RAS}$ i $\overline{CAS}$ aktivirani, a $\overline{WR}$ neaktivan (na logičkoj jedinici) memorijska kola će ovo shvatiti kao čitenje i podatke sa adresirane lokacije ća postaviti na magistrali podataka.

Za vreme osvežavanja memorije aktivira se samo $\overline{MREQ}$, odnosno $\overline{RAS}$, dok su ostali signali naaktivni. Na adrasna linije kola se dovoda adrese A0-A6 (S na logičkoj jedinici) koje su potrebne za osvažavanja memorije.



Slika 6-7 32K RAM

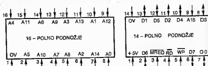
Kod verzije Spekturma 6a primenjano ja kolo ZX8401 koje zamanjuje sva četiri multipleksera 74LS157 i kolo za formiranje $\overline{RAS}$ i $\overline{CAS}$ signala za 32K RAM.

PROŠIRIVANJE MEMORIJE 16K SPEKTRUMA

Ovde je opisan način prepravljanja Spaktruma sa 16K RAM-a u 48K Spektrum.

Verzija 1

Na ploči ove verzije Spektruma postoje dva podnožja za proširenje mamorije, jedno sa 14 i drugo sa 16 izvoda. Najlakše je kupiti gotovu štampanu ploču sa već ugrađenim elementima i priključiti je na ova podnožja. Oni koji žele mogu prema postojećoj šemi 32K RAM-a da naprave ovu pločicu. Na slici 6-8 je dat raspored signala na izvodima podnožja za proširenje.



Slika 6–8 Podnožja za proširivanje memorije u Spektrumu verzije 1

Verzija 2

Ova verzija ima na štampanoj ploči predviđena mesta i podnožja za potrebna integrisana kola koja samo treba tačno ubaciti (slika 6–9). Treba paziti da udubljenja za identifikaciju na integrisanim kolima budu okrenuta u istom smeru u kome su označena na crtežu.

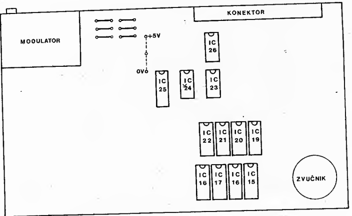
Spisak potrebnih elemenata:

IC15–IC22 8 memorijskih kola: TI 4532-3 ili TI 4532-4 sva moraju da budu istoga tipa

IC23 74LS32

IC24 74LS00

IC25–IC26 74LS157 (ne mogu da budu firme NATIONAL SEMICONDUCTOR)

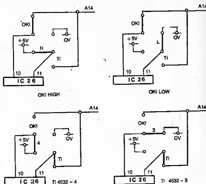


Slika 6–9 Položaj integrisanih kola potrebnih za proširivanje memorija Spektruma verzije 2 i 3

TI 4532 su memorije proizvođeče Texas Instruments kapaciteta 64K x 1 bit koje imaju u jednoj polovini memorijskog prostora nekoliko neispravnih lokacija tako da se koristi samo ispravna polovina. 4532-3 ima ispravan gornji deo, a 4532-4 donji deo adresnog prostora Na štampanu ploču trebe zalemiti perče žice (engl. link) kojim je određeno koja se polovina upotrebljava Za TI 4532-3 povezati srednju tačku sa tačkom označenom sa +5V (link 3), a za TI 4532-4 povezati srednju tačku sa tačkom označenom OV (link 4).

Verzije 3

U ovom slučaju IC15-IC22 pored TI 4532-3 ili TI 4532-4 mogu biti i OKI MSM 3732 200ns (250ns). Sva kola moraju da budu istog tipa. Ostala četiri kola su ista kao u prethodnom slučaju.

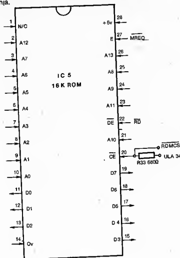


Slika 6–10 Potrebne veze u zavisnosti od vrste upotrebljene memorije pri proširivanju memorije Spektruma verzije 3

Pomoću dva kretke provodnike (žice) potrebno je povezati nekoliko tačaka ne predviđenim mestime, u zavisnosti od upotrebljenih memorijskih kole. Raspored je det na slici 6-10.

6-3 ROM

U ROM memoriju (reed only memory) podaci se upisuju u toku proizvodnje i više se ne mogu menjati. Ne brišu se ni nestankom napajenja.



Slika 6–11 ROM

Spaktrumov ROM je integrisano kolo u pakovanju sa 28 izvoda, kapaciteta 16K bajta. Proizvođači su HITACHI ili NEC. U njega su upisani sistamski programi (oni koji omogućavaju korišćenja računara): operativni sistem i bejzik interpretar. ROM zauzima adrese od 0 do 16383 što je najzgodnije jer Z80 po uključivanju počinje izvršavanja programa od adrasa 0.

Funkcije izvoda su sledeće:

-D0-D7, osam izlaza, magistrala podataka.

Prako njih podaci iz adrasirana mamorijske lokacije dospevaju prako magistrale podataka, u cantralnu procesorsku jadinicu.

-A0-A13,14 ulaza, adrasne linija

Prako njih mikroprocesor adresira odrađanu mamorijsku lokaciju.

-$\overline{CE}$ (chip anabla), ulaz za aktiviranje kola

Povezan je preko otpornika od 680 oma sa odgovarajućim izlazom na ULA-l koji se aktivira kada sa podaci čitaju iz ROM-a, odnosno kada su adresne linije A14 i A15 na logičkoj null.

-$\overline{E}$ (enable), ulaz koji takođa služi za aktiviranja kola. Povezan je sa$\overline{MREQ}$ izlazom na CPU prako koga se signalizira da sa komunikacija vrši sa mamorijom, a na sa ulazno-izlaznom jadinicom.

-$\overline{OE}$ (output enabla) ulaz, aktiviranja izlaza.

Omogućava da podaci preko kola za vezu iz adresirane lokacije dospeju na magistralu podataka. Povezan je sa$\overline{RD}$ na CPU.

-$\overline{CE}$ izvod ja povezan sa$\overline{ROMCS}$ (ROM chip salect) priključkom na konaktoru koji se nalazi na zadnjoj strani Spektruma. Kada se on veže za +5V, ULA više ne može da aktivira ROM (otpornik od 680 oma dozvoljava ovu mogućnost jer sprečava preopterećenje izlaza ULA-e ovim kratkim spojem). Na ovaj način možemo da priključimo neku drugu mamoriju, RAM, ROM ili EPROM koja će biti adresirana u ovom adresnom prostoru.

EPROM (Erasable Programable ROM) mamorije su vrsta ROM mamorije koje sa ne programiraju u toku proizvodnje već kasnije pomoću uređaja, programatora EPROM-a. Na sradini njihovog pakovanja se nalazi staklani prozor kroz koji moža da buda osvatljena ultraljubičastom svatlošću, čime se podaci brišu, pa se mogu upisati novi. Nestankom napajanja podaci sa ne brišu.

EPROM tipa 27128 ima potpuno isti kapacitet i raspored izvoda kao Spektrumov ROM i mogao bi se ubaciti umesto njega. Uključivanjem napajanja počeo bi da se izvršava program koji se nalazi u njamu.

Zanimljiva mogućnost je upisivanje sadržaja ROM-a u EPROM, s time da se u 1K neiskorišćenog prostora upiše neki korisni program.

6–4 ULA

ULA (Uncomitted Logic Array – skup logike čije se funkcija bira preme potrebi) je integrisano kolo se 40 izvoda koje zamenjuje veliki broj menjih logičkih kole. Napravljeno je specijalno za Spektrum. Proizvodač je firma Ferranti. Do sada je u Spektrume ugradivano nekoliko verzija ovih kola. Pomoću ULA je smanjena cena i veličina računara. Ona ima funkciju de komunicira sa ulazno-izlaznim jedinicama. Ulazne su tastatura i kasetofon, a izlazne TV ekran, zvučnik i kasetofon.

Funkcija pojedinih izvoda

–D0-D7, osam ulaza-izlaza, megistrale podateka

–A14-A15, ulazi, linije adresne meglstrale

Na osnovu njih ULA prepoznaje sa kojim delom memorije se komunicira. ROM–a (adrese od 0 do 16383), prvih 16K RAM-a (16384-32767) ili sledećih 32K RAM-a (32767-65535).

–XTAL, ulaz Za ovaj izvod je vezan kristal kvarca koji kontroliše frekvenciju oscilatora od 14MHz koji se nalazi u ULA-i. Deljenjem ove učestenosti sa dva dobija se 7MHz potrebnih za generisanje slike, a deljenjem još sa dva dobija se signal frekvencije 3,5MHz koji se koristi kao klok mikroprocesora.

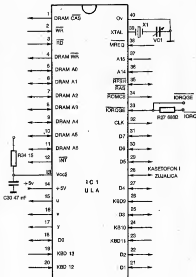
–CLK, izlaz,daje klok frekvencije 3,5MHz za centralnu procesorsku jedinicu.

–$\overline{WR}$, ulaz, povezan sa odgovarajućim izlazom mikroprocesora. Kade je aktivan, mikroprocesor upisuje podatek u memoriju ili ulazno-izlaznu jedinicu.

–$\overline{RD}$, ulaz, povezan sa odgovarajučim izlazom mikroprocesora. Aktivan je kade mikroprocesor čita podatak iz memorije ili ulazno-izlazne jedinice.

–$\overline{MREQ}$, ulaz, vezen ze odgovarajuči izlaz mikroprocesore. Aktivan je kada mikroprocesor komunicira sa memorijom.

– $\overline{IORQGE}$, ulaz, vezen za $\overline{IORQ}$ izlaz mikroprocesora preko otpornike od 680 oma (R27) i povezan sa priključcima za konektor na zadnjoj strani Spektruma. Ako se veže za +5V sprečave se da mikroprocesor komunicira sa ulazno-izlaznim jedinicama preko ULA i



Slika 6–12 ULA

omogućeva da se umesto toga priključe druge ulazno-izlazne jedinice

– $\overline{ROMCS}$, izlaz, vezan za $\overline{CS}$ ulaz ROM-a preko otpornika od 680 oma (R33). ULA ga aktivira kade su A14 i A15 na logičkoj nuli, tj. keda se podaci čitaju iz ROM-a. $\overline{CS}$ je vezan za $\overline{ROMCS}$ priključak za konektor ne zadnjoj strani Spektruma. Kada se važe za +5V sprečave se selekcija (ektiviranje) ROM-a i omogućava priključenje druga memorija u tom adresom prostoru.

–$\overline{INT}$, izlaz priključen na odgovarajući ulaz na mikroprocesoru. ULA ga aktivira svakih 20 ms i time izaziva prekid programa koji mikroprocesor izvršava i prelazak na program prakida. Tej program očitava stanje na tastaturi i povačava za jedan sadržaj sistemske promenljive FRAMES.

–KBD-9 do KBD-13, pet ulaza sa tastature

–$\overline{RAS}$ (row eddress strobe), izlaz koji je ektivan kada prvih 16K RAM-a treba de učitaju adrese reda (A0-A6). Preko otpornika od 330 oma (R57) vezan je za $\overline{RFSH}$ izlaz mikroprocesore tako da on može da osvežava memoriju u toku impulsa za vertikalnu sinhronizaciju slike koji traje 5 ms. Inače ovu memoriju osvežava ULA čitenjem podatake za sliku.

–$\overline{DRAM\ CAS}$ (coulmn addrass strobe), izlaz koji je aktivan kada prvih 16K RAM-a treba da učitaju edresu kolone (A7-A13)

–U, vidao izlaz boje, razlika između intenziteta plavog i žutog

–V, video izlaz boje, razlika između intenziteta crvenog i žutog

–Y, video izlaz, luminentni i sinhronizacija

Ova tri izlaza sa vode na video moduletor LM 1889

–Ulaz, izlaz ze kasetofon i zujalicu

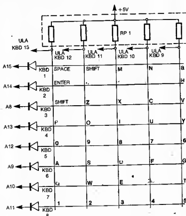
–$\overline{DRAM\ WR}$ izlaz vezan za $\overline{WR}$ ulaz preko 16K RAM-a. Komande upisa u ovu memoriju.

–DRAM A0–DRAM A6, sedam ulaze-izlaza vazanih za odgovarajuće adrasne priključke prvih 16K RAM-a. Preko njih ULA dobija informacije o adresi koju bira mikroprocesor, e keda osvežava sliku njima adresira deo memorije u kome se nalaze potrabne informacije. Tada ima prednost nad mikroprocesorom zbog toga što se adrasa se njega dovodi preko otpornika od 330 oma.

6–5 TASTATURA

Spektrum ima 40 tastere koji su međusobno povazani provodnicima. Postoji osam provodnika od kojih svaki povezuja pet tastera

koji su u istom redu i pet provodnika koji povezuju po osam tastera koji su u istoj koloni. Jedan izvod svakog tastara je vezan za jedan provodnik reda, a drugi za jedan provodnik kolone. Svaka kolona je preko otpornika vazana za +5V, pa je to napon na njoj kada nije aktivirana, odnosno kada nijedan taster vezan za nju nije pritisnut. Provodnici svakog rada su diodama povezani za po jednu adresnu liniju od A8 do A15.



Slika 6–13 Tastatura

Informacija o tome da li je neki taster pritisnut dobija se tako što mikroprocesor sve adresne linije od A0 do A15 postavlja ne nivo logičke jedinice, sem jedne koju drži na nivou logičke nule. Zatim učitava podatke sa provodnika kolone preko izvoda KBD 9 do KBD 13 na ULA. Ukoliko nijedan taster koji je povezan sa linijom koja je na logičkoj nuli nije pritisnut, svi provodnici kolone su na nivou logičke jedinice, pa se ispitivanje nastavlja. Sledeće linija reda se postavlja ne logičku nulu, e ostale na logičku jedinicu i opet se učiteva stanje na linijema kolona i tako redom, dok se ne ispita svih osam redova. Kada je jedan taster pritisnut, a prilikom ispitivanja se provodnik reda za koji je on vezan postavi na nivo logičke nule, preko njega će i linija kolone za koju je on vezan morati da se spusti na nivo logičke nule. Računar će učitati ovaj podatak i na osnovu aktiviranog reda i aktivirane kolone utvrditi o kom tasteru se radi. Spektrum ispituje stanje na tastaturi svakih 20 ms u toku izvršavanja programa prekida

Izvodi KBD-9 do KBD-13 ULA-e su deo ulaza (D0-D4) osmobitne ulazne jedinice računara sa adresom 254. Podaci sa nje se dobijaju izvršavanjem naredbe **IN 254**. Pri tome će svaki bit dobijenog podatka biti jednak logičkom nivou na jednoj liniji kolone i to:

D0-logički nivo na	KBD-13
D1-	KBD-12
D2-	KBD-11
D3-	KBD-10
D4-	KBD-9

D5-neupotrebljen

D6-logički nivo na ulazu sa kasetofona (EAR)

D7-naupotrebljen

Ako želimo, možemo da upotrebimo sledeće naredbe de bismo učitali stanje ne pojedinim redovima, jer one istovremeno jednu adresnu liniju od A8 do A15 postavljaju na logičku nulu, a ostale na logičku jedinicu.

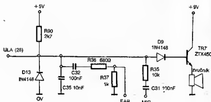
naredba	red koji se učitava	linija ne logičkoj nuli
IN 32766	SPACE do B	A15
IN 4915	ENTER do H	A14
IN 57342	P do Y	A13
IN 61438	0 do 6	A12
IN 63486	1 do 5	A11

IN 64510 Q do T A10
IN 65022 A do G A9
IN 65278 SHIFT do V A8

PRIMER: ako naredbom **IN** 64510 učitamo bajt u kome je bit D2=0 to znači da je pritisnut taster E.

6–6 KASETOFON

Signal sa kasetofona dolazi preko otpornika od 680 oma i kondenzatora od 100 nF na izvod 28 ULA. Informaciju o njemu daje bit D6 ulazne jedinice sa adresom 254. Ukoliko je taj signal dovoljne amplitude, moći će vrlo tiho da se čuje u zvučniku Spektruma. Dioda D13 uklanja suviše velike negativne napone koji tom prilikom mogu da se jave.



Slika 6–14 Kolo zvučnika i kasetofona

Prilikom snimanja podataka na traku mikroprocesor aktivira bit D3 izlazne jedinice koja ima adresu 254. Tada signal iz izvoda 28 ULA preko otpornika od 10K (R35) i kondenzatora od 100 nF (C31) dolazi na ulaz za mikrofon kasetofona. Njegova amplituda ja 1.3V. Za isti izvod ULA vezan je zvučnik preko diode i tranzistora. Pad napona na diodi koja provodi tranzistoru je 1,2V, tako da ovaj signal nije dovoljan da aktivira zvučnik.

6–7 ZVUČNIK

Spektrum nema klasični zvučnik već elektroakustički pretvarač u obliku okrugle pločice. Smešten je na štampanoj ploči sa donja desne strana. Napon koji se dovodi na pratvarač dovodi do kretanja njegova mambrane, tako da se električne oscilacija pratvaraju u akustičke. Ovakvi pretvarači su vaoma česti u časovnicima i drugim električnim uređajima sa alarmom.

Zvučnik se aktivira bitom D4 izlazna jedinice čija ja adresa 254. Tom prilikom se na izvodu 28 ULA-a dobija napon od 3,3V koji je dovoljan da aktivira diodu i tranzistor preko kojih je zvučnik vezan za ovaj izvod (oni sprečavaju da se u njemu čuja signal prilikom snimanja programa na traku).

Signal koji čujemo u zvučniku dolazi i do izlaza MIC tako da ga možemo snimiti na magnatofon ili pojačati i čuti preko zvučnika.

U verzijama Spektruma 1 i 2 zvučnik je za ULA vazan preko dve dioda, dok je u kasnijim verzijama umesto druge diode upotrebljen trenzistor, čime je struja kroz zvučnik povećana, pa je zvuk postao jači.

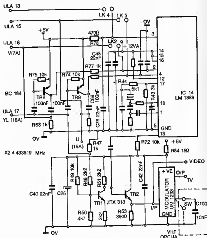
6–8 VIDEO IZLAZ

ULA proizvodi tri video signala: Y je mešavina luminentnog (osvetljanost) i impulsa za sinhronizaciju slike, U je informacija o razlici intenzitata plave i žute boje, a V o razlici intenzitata crvana i žute boja. Padaset puta u sekundi ona čita sadržaj vidao memorije čija ja adresa od 16384 do 22527 (engl. display file) i na osnovu tih podataka stvara ove signale pomoću kojih se na akranu dobija slika. Ovolika učestanost omogućava da se usled inercije oka na primećuje treperenje. Ako mikroprocesor pokuša da komunicira sa video memorijom dok se osvežava slika, ULA ga zaustavlja ukidanjem kloka. To se na njegov rad odražava samo kao vrlo kratka pauza.

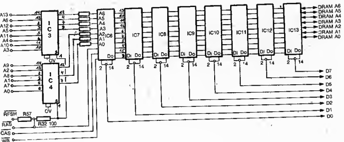
Boja oboda slika je određena bitima DØ, D1 i D2 izlazne jedinice čija je adrasa 254. Ako svi sadrža logičku nulu, boja ća biti crna, a ako svi sadrže logičku jedinicu dobiće se bela, dok se kombinacijama dobijaju ostale boja.

Integrisano kolo LM 1889 stvara složeni signal boje. U njamu se nalazi oscilator kontrolisan kristalom kvarca X2. Signal koji on proizvodi moduliša se pomoću U i V. Dobijeni složeni signal se meša sa Y pomoću tranzistora TR1 i TR2 i dobija se kompozitni kolor

video signal. Njime se u modulatoru moduliše signal čija učestanost odgovara 36. kanalu televizije, čime se omogućava dobijanje slike ne standardnom TV prijemniku. Bolji kvalitet slike se dobija ako se kompozitni kolor video signal bez modulacije odvede na TV monitor.



Slika 6–15 Video kolo



Slika 6–6 16 K RAM

Moguće je i televizor koristiti kao monitor ako mu se video signal dovede posle kola za demodulaciju slike.

Kod Spektruma verzije 1 i 2 ponekad se dešava de frakvencija signale boje ne odgovare onoj u TV prijemniku, što onemogućava dobijanje slike u boji. U tom slučaju je moguće pomoću promenljivog kondenzatora VC2 menjati njegovu učestanost sve dok se boje ne pojave. Sadržaj boje se može podešavati trimer potenciometrima VR1 i VR2. VR1 utiče na crvenu i žutu, a VR2 na plavu i žutu boju. Ne pruge po ivicama figura i karaktera ne ekranu moguće je uticati promenom frekvencije oscilatora na ULA, što se postiže podešavanjem pomoću promenljivog kondenzetora VC1 (komponente su obeležene ne štampanoj ploči).

Kod Spektrume verzije 3 i novijih, VR1 i VR2 ne postoje a njihovu funkciju obavljaju tranzistori TR8 i TR9, teko da nema potrebe za podešavanjem. Takođe su i promanljivi kondanzatori zamenjeni običnim, nepromenljivim, jer su upotrebljeni kristali veće tačnosti.

6 – 9 NAPAJANJE

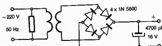
Kolo za napajanje Spektruma je podeljeno u dva dela: jedan čini mrežno napajanje (ispravljač) koji se nalazi u posebnoj kutiji, a drugi daja napon od 5V, 12V i -5V i nalazi se u kutiji računara

MREŽNO NAPAJANJE (ISPRAVLJAČ)

Mrežni ispravljač se priključuje na mrežni napon 220V, 50Hz, a na izlazu daja neregulisani napon od 9V. Maksimalna izlazna struja je 1,4A. Sastoji se od mrežnog transformatora koji izoluje ostali deo kole od mrežnog napona i daje potrebni neizmenični napon. On se ispravlje sa četiri diode vezane u Grecov spoj. Kondenzator filtrira dobijeni jednosmerni napon.

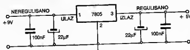
NAPAJANJE +5V

+5V se dobija pomoću integrisanog kola-regulatora 7805 koji se napaja sa +9V iz mrežnog ispravljeća. Integrisano kolo je redi hledenje pričvršćeno za aluminijumski lim. Dobijeni napon je veoma stabilan sve dok ne dođe do preopterećenja kola. Ono je predviđeno da obezbedi struju do 1A. Regulator je zaštićen od preopterećenja, tako de pri strujama većim od 1A počinje da pada napon na izlazu. Kondenzatori C2 i C3 filtriraju izlazni napon i sprečavaju oscilovanje regulatore. C3 se upotrebljava iako ima kapacitet manji od C2



Slika 6–16 Mrežno napajanje – ispravljač

zato što ja njegova impedansa mnogo manja na višim frekvencijama. Ovaj napon služi za napajanje logičkih kola. Verzija Spektruma od 48K RAM-a troši oko 1A što je na granici mogućnosti regulatora, pa se pri priključivanju novih periferija preporučuje dodavanje još jednog. Ovo je opisano u delu knjige o projektima.

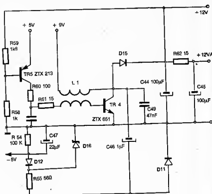


Slika 6–17 Napajanje + 5V

NAPAJANJA + 12V i –5V

Ovi naponi su potrabni za napajanje 16K RAM-a, a +12VA služi za napajanje LM1889. Dobijanja ovakvih napona iz +5V i +9V moguće je zahvaljujući oscilatoru koga sačinjavaju tranzistor TR4, transformator L1, kondenzator C43 i otpornik R61. Napon na kolektoru TR4 se menja između 0 i 13V. Preko diode D15 puni se kondenzator C44 i tako se formira napon od 12V (pad napona na diodi je 0,6V). Njegovu veličinu reguliše povratna veza preko otpornika R59 na tranzistor TR5. Ukoliko napon na izlazu počinje da se smanjuje, TR5 će davati vaću struju. Zbog toga će se povećati frekvencija oscilatore i napon na izlazu će porasti. +12VA se dobija iz 12V filtriranjem preko otpornika i kondenzatora.

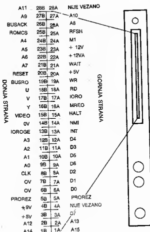
–5V se dobija preko kondenzatora C46 koji se puni kroz D11 kada je napon na kolektoru pozitivan, a prazni kroz diodu D12 i kondenzator C47 kada je napon na kolektoru oko 0V. Otpornik R65 i zener dioda D17 stabiliziraju napon na –5V.



Slika 6–18 Napajanje + 12V i – 5V

KONEKTOR ZA PRIKLJUČIVANJE DODATNIH UREĐAJA

Na zadnjoj strani Spektruma sa nalazi konektor za priključivanje dodatnih uređaja, koji se preko njega povezuju sa magistralom podataka, adresnom magistralom, kontrolnim linijama i linijama napajanja. Raspored signala na pojedinim izvodima konektora je dat na slici.



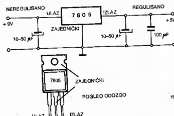
Slika 6–19 Konektor za priključivanje dodatnih uređaja

7

Projekti

7–1 DODATNO NAPAJANJE

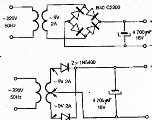
Perifernim jedinicama koje se priključuju na Spektrum uglavnom je potrebno napajanje od +5V. Ono se može obezbediti priključivanjem na odgovarajući izvod na sistemskom konektoru koji se nalazi ne Spektrumu. U tom slučaju se opterećuje regulator 7805 koji se nalazi u računaru. Kod 48K Spektrume on već radi na granicama svojih mogućnosti pa se preporučuje dodavanje spoljnog regulatora.



Slika 7–1 Dodatno napajanje +5V i raspored izvoda regulatora 7805

Šema je potpuno ista kao i ona koja se već primenjuje u samom Spektrumu. Ovo kolo se priključuje na +9V iz Spektrumovog ispravljača. Najzgodnije je priključiti ge na izvod 4B ne konektoru. Na ovaj način se dobija napajanje od +5V sa maksimelnom strujom od 400 mA. Pri većim strujama će doći do preopterećenja ispravljača. Regulator 7805 treba pričvrstiti na hlednjak, koji može biti od aluminijumskog lima. Izvodi ovog integrisanog kola su dati na slici.

Keda je potrebno de struja napajanje bude veća, dodaje se snažnije nepajanja. Ono mora da obezbedi neregulisani napon od 9V, a njegova struja je jednaka zbiru one koja je potrebne za napajanje Spektruma (1A u slučaju 48K verzije) i one koja je potrebna za napajanje perifernih jadinice. Prikazene su dve verzije



Slika 7–2 Dve verzije dodatnog mrežnog napajanja

Šema prve je potpuno ista kao ona koju koristi Spektrum. Jedina razlike je u toma što transformator i diode mogu da obezbede veću struju (ne primer 2A). Druga verzija koristi transformator čiji sekundar ime srednji izvod. Napon između srednjag izvoda i svekog od krajnjih je 9V. Transformator i diode treba de budu izabrani tako da mogu da obezbede potrebnu struju. Elementi naznačeni na slici odgovaraju struji od 2A.

Pre nego što se napravljeno mrežno napajenje priključi na računar, treba ispitati njegov napon kako ne bi došlo do oštećenja ili uništenje računara. Priključenje je mogućno prako utičnice označene sa 9VDC ili preko izvode na konektoru (+9V na 4B, a 0 ne 6B ili 7B). Greška u mestu priključivenje i polaritetu mogu de imaju katastrofalne posledice, što je i provereno.

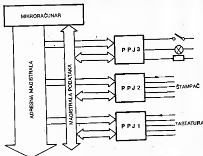
Mogućno je koristiti Spektrumovo napejanje ze napajanje samog Spektruma, a dodatno napajanje za periferna jedinice. U tom slučaju je potrebno paziti da se naponi na izlazu jednog i drugog napajanja pojavaljuju istrovemeno. Kada bi se prvo pojavio semo jedan napon, mogeo bi da uništi kola onog dele koji još nije dobio napajanje. Zato se i preporučuje napajanje iz istog ispravlječa. Prednost

korišćenja i Spektrumovog napajanja je u tome što su potrebni transformator i diode koji obezbeduju manju struju.

Na perifernim jedinicama treba između linija napajanja od +5V i 0V stavljati kondenzatore od 100 nF radi smanjenja smetnji. Potreban je po jedan kondenzator na jadno ili dva integrisana kola, a treba da budu postavljeni što bliže izvodima za napajanje kola.

7–2 PARALELNE PERIFERNE JEDINICE

Paralelne periferne jedinice su univerzelna kola namenjena povezivanju raznih ulazno-izlaznih uređaja sa računarom (kao što je Spektrum na primer). Zovu se paralelne zato što postoji mogućnost da se istovremeno izvrši prenos više bita informacija (najčešće 8 bita-bajt).
Ne slici je prikazana upotreba tri paralelne periferne jedinice (PPJ). Podaci sa razmenjuju preko magistrale podataka pod nadzorom kontrolnih linija $\overline{IORQ}$,$\overline{RD}$,$\overline{WR}$...). Stanje na adresnoj magistrali određuje sa kojom jedinicom se komunicira



Slika 7–3 Primer upotrebe paralelne periferne jedinice

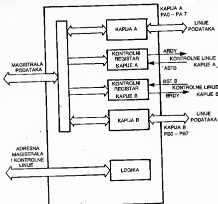
PPJ1 je ulazna jedinica preko koje se dobijaju podaci sa tastature, PPJ2 je izlazna jedinica preko koje se podaci šalju na štampač

a PPJ3 je izlazna i ulazna jedinica preko koje se dobije informacije o stanju prekidače, daju komande za paljenje sijalice i aktiviranje relea.

Engleski nazivi koji se najčešće koriste ze paralelne periferne jedinice su: PIO–parallel input output (parelelni ulaz izlaz), PIA–parallel interfece edapter (paralelni adapter za vezu), PPI–parallal peripheral interface (paralelna pariferijske veza)

Z80A PIO

Z80A PIO je parelelna ulazno-izlazne jadinica koja je namenjene povezivanju mikroračunarskih sisteme zesnovanih na Z80A CPU, sa periferním uređajime. Podaci se razmenjuju preko dve osmobitne kapije (eng. port) koje mogu biti ulazne, izlazne i istovremano i ulazne i izlazne. Kapiju je najjadnostavnije posmatrati kao kolo koje se sastoji od ulaznog registra (register-osam memorijskih ćelija), izlaz-



Slika 7–4 Blok šema Z80 PIO

nog registra i dodatne logike. Oba registra se aktiviraju istom adresom, ali prvi samo pri čitanju naredbom **OUT**, a drugi samo pri upisivanju podataka naredbom **IN.** Svaka kapija ima za povezivanja sa perifernom jedinicom pored osam linija podataka i dve kontrolne linije kojima se kontroliše međusobna komunikacija (tzv. handshake--hendšejk linije). Postoje i dva osmobitna kontrolna registra. Podaci koji se u njima nalaze definišu funkciju koju kapije imaju.

Ostali deo Z80 PIO čina kola za kontrolu i povezivanja sa magistralama. Ona omogućavaju i vrlo efikasnu razmenu podataka sa Z80 CPU pomoću prekida (interrupt), ali o tome, zbog složenosti, ovda naće biti reči.

Za programera je bitno da mikroprocesor „vidi" ovu jedinicu kao skup od četiri registra, odnosno četiri memorijske lokacije u adresnom prostoru ulazno-izlanih jedinica.

Dve su za podatke, a dve za kontrolu.

Kapija mogu da rade na četiri različita načina:

način0 (moda 0) – izlaz

način 1 (mode 1) – ulaz

način 2 (mode 2) – dvosmerni, samo kapija A ima tu mogućnost

način 3 (mode 3) – kontrolni, pojedine linije su ulazi, a pojedine izlazi

Moguća je da kapija A radi na jedan način, a kapija B na drugi. Izuzetak je samo kad je kapija A dvosmarna. Tada kapija B može da radi samo na kontrolni način (3).

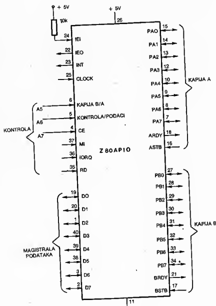
Z80A PIO je integrisano kolo sa 40 izvoda.

Funkcije pojedinih izvoda su:

– D0 –D7, ulazno-izlazne linije, magistrala podataka, mogu da budu u stanju visoke impedanse

–$\overline{CE}$ (chip enable), ulaz, logička nula na ovom izvodu omogućava razmenu podataka i komandi između PIO i mikroprocesora (služi za adresiranje). Izvodi B/A i C/D su ulazi koji određuju sa kojim registrima ća mikroprocesor komunicirati.

B/A	C/D	REGISTAR SA KOJIM SE KOMUNICIRA
0	0	kapija A, podaci
0	1	kontrolni registar kapije A
1	0	kapija B, podaci
1	1	kontrolni ragistar kapije B



Slika 7–5 Izvodi i način povezivanja Z80 PIO

– ф, ulaz, klok Z80A sistema
– $\overline{RD}$ (read), ulaz, označava da mikroprocesor čita podatak iz memorije ili ulazno-izlazne jedinice.
– $\overline{M1}$, ulaz, povezuje se sa istoimenim izlazom mikroprocesora Služi za sinhronizeciju operacija u PIO i za indikaciju na sledeći način: kada su aktivni $\overline{M1}$ i $\overline{IORQ}$ mikroprocesor potvrđuje prekid, e kada je $\overline{M1}$ aktivan, a $\overline{RD}$ i $\overline{IORQ}$ neaktivni, PIO se resetuje.
–$\overline{IORQ}$ (input output request), ulaz, vezen za odgovarajući izlaz CPU.
Logičkom nulom označava de se komunikecija vrši sa perifernom jedinicom, a ne sa memorijom. Kada je istovremeno ektivan i $\overline{RD}$ podatak se čita iz periferne jedinice, a kada je $\overline{RD}$ neaktivan (logička jedinica) upisuje se u PIO.
– PA0 –PA7, ulazi ili izlazi kapije A, mogu da budu i u stanju visoke impedanse. Služe za prenos podataka između kapije A i perifernog uređaja.
– $\overline{A\ STB}$ (port A strobe), ulaz, kontrolni (hendšejking) signal kapije A
– A RDY (register A ready), izlaz, kontrolni (hendšejking) signal kapije A
– PB0 –PB7, ulazi ili izlazi kepije B, mogu da budu i u stanju visoke impedanse. Služe za prenos podataka između PIO (kapije B) i perifernog uređaja. Mogu da obezbede struju od 1,5 mA pri naponu od 1,5 V, što je dovoljno za pobudu darlington tranzistora.
– $\overline{B\ STB}$ (port B strobe), ulaz, kontrolni signal kapije B
– B RDY (register B ready), izlaz, kontrolni signal kapije B
–$\overline{INT}$ (interrupt request), izlaz sa otvorenim drejnom (struja može samo da ulazi u njega).
Aktiven je kada PIO traži prekid od centralne procesorske jedinice.
– IEI (interrupt enable in), ulaz
Logičke jedinica na ovom ulazu dozvoljava da PIO generiše prekid, jer to znači da nijedne jedinica višeg prioritete trenutno ne traži prekid.
– IEO (interrupt eneble out), izlaz
Ovaj izlaz je na logičkoj jedinici samo keda je ulaz IEI na logičkoj jedinici i kada PIO ne traži prekid. Služi da signalizira perifernim jedinicama nižeg prioriteta da mogu da traže prekid logičkom jedinicom.

PROGRAMIRANJE PIO

RESET

Z80 PIO se resetuje automatski uključivanjem napajanja. Drugi način je ektivirenjem signala $\overline{M1}$ bez aktiviranja $\overline{RD}$ i $\overline{IORQ}$ signala.

Keda se PIO resetuje automatski se prelazi u način rade 1 (mode 1) kada su kapije ulazi. Oba izlazna registra su resetovana.

PROGRAMIRANJE

Način rade kapije se bire upisivanjem jednog bajte, koji sa naziva kontrolne reč, u osmobitni kontrolni registar te kapije.

D7 D6 D5 D4 D3 D2 D1 DØ
M1 MØ x x 1 1 1 1 x neupotrabljen bit

Nečin označeva da se
rada bira način rade

Biti DØ do D3 kontrolna rači moreju da budu 1, jer se time označava da se pomoću nje bira način rada kapije. Postoja i druge funkcije kontrolne reči. Vrednost bite D4 i D5 nije bitne, a vrednost bita D6 i D7 određuje način rada na sledaći način:

D7	D6	način rada
Ø	Ø	Ø – izlaz
Ø	1	1 – ulaz
1	Ø	2 – dvosmerno, samo A
1	1	3 – kontrolni

Način Ø – Izlaz

Izlazni register je aktivan, a ulazni neektivan. Podetek koji se upisuje u kapiju naredbom **OUT,** pamti se u njenom izlaznom registru i pojavljuje ne izvodime ze podatke (PAØ–PA7 ili PBØ–PB7). Sadržaj izlaznog registra se menja upisivanjem novog podatka. Podetak koji sa u njemu nalazi može sa učitati u mikroprocesor naredbom **IN.**

Kada se podatak upiše, kontrolne linija RDY (A ili B) se postavlja na nivo logičke jedinice da bi naznačila perifernom uređeju de su mu podaci dostupni. Prijem podataka periferni uređej signallzire slanjem impulsa kroz kontrolnu linijuSTB (A ili B). Tada se kontrolna linije RDY vreća na nivo logičke nule. Pozitivna ivice signaleSTB generiše signal prekida.

Način 1 – ulaz

Izlazni register je neektivan, a ulazni je aktiven. Kada je signal RDY ne logičkoj jedinici periferni uređaj postav'ja podetke na linije podateka (PAØ–PA7 ili PBØ–PB7) i obare signalSTB na logičku nulu. Podaci se upisuju u ulazni registar kapije i signal RDY se vraća na nivo logičke nule. Istovremeno se generiše signal prekida. Mikrop-

rocesor čita podatak iz PIO naredbom IN. Izvršavanje te naredbe dovodi do prelaska linije RDY na logičku jedinicu čime se periferiji dovodi do prelaska linije RDY na logičku jedinicu čime se perifernom uređaju signalizira da može da upiše novi podatak.

Način 2 – Dvosmerni

Kapija A je i ulazna i izlazna, dok se kapija B ne upotrebljava ili je u kontrolnom načinu rada. Kontrolne linije kapije A se koriste za izlazne operacije.

Kada jeA STB na nivou logičke nule, podaci iz izlaznog registra kapije A se nalaze na linijama podataka PA0–PA7. Ako jeA STB na nivou logičke jedinice, podaci mogu da se upišu u ulazni registar kapije A aktiviranjem linijeB STB. Signali A RDY i B RDY mogu da budu aktivni u isto vreme označavajući i da su izlazni podaci sa PIO dostupni (A RDY) i da je PIO spremna da primi podatke sa perifernog uređaja (B RDY).

Način 3 – kontrolni

Neke linije kapije koja radi na ovaj način mogu da budu ulazne, a neke izlazne. Koje su ulazne, a koje izlazne određuje se slanjem još jedne kontrolne reči, posle one koja bira ovaj način rada.

D7 D6 D5 D4 D3 D2 D1 D0

Ako je neki bit te druge kontrolne reči na jedinici odgovarajuća linija podataka će biti ulazna, a ako je nula, odgovarajuća linija podataka će biti izlazna.

STB signali nemaju uticaja, a RDY signali su uvek na logičkoj nuli. Podaci mogu da se čitaju ili upisuju bilo kada.

Upisivanje podataka u kapiju će uticati samo na izlazne linije. Podaci koji se čitaju sa kapije će se sastojati od ulaznih podataka sa linija koje su ulazi i podataka iz izlaznog registra onih linija koje su izlazi.

Povezivanje Z80 PIO sa Spektrumom

Z80 PIO se može povezati sa Spektrumom na sledeći način: izvodi od D0 do D7 se povezuju sa odgovarajućim linijama magistrale podataka, izvodiM1,IORQ,RD i CLOCK sa odgovarajućim izvodima na konektoru, izvod 11 je 0V, a 26 napajanje od +5V. Prekid sada načemo koristiti pa sa ulaz IEI (izvod 24) vezuje na +5V, a izlaz IEO, INT se ostavljaju slobodni. pri komuniciranju sa ugrađenim ulazno-izlaznim jedinicama Spektrum ne koristi adresne linije A5, A6 i A7 (drži ih na logičkoj jedinici pa će zato one biti upotrebljene. A7 se vezuje naCE tako da logička nula na ovoj liniji omogućuje da mikroprocesor komunicira sa PIO. A6 je povezan sa izvodom 5 pomoću

koga se biraju kontrolni registri (A6 na logičkoj jedinici) ili registrima podataka (A6 na logičkoj nuli). A5 preko izvoda 6 bira da li se komunicira sa registrima kapije A (A5 na logičkoj nuli) ili kapije B (A5 na logičkoj jedinici). Adresa PIO trebe da bude takva da ostale adresne linije (A0–A4) budu na logičkoj jedinici, kako se ne bi aktivirala istovremeno neka druga ulazno-izlazna jedinica, pa se dobija:

Izabreno	Adresa binarna	Decimalna	Heks
kapija A – podaci	00011111	31	1F
kontrolni registar kapije A	01011111	95	5F
kapija B – podaci	00111111	63	3F
kontrolni registar kapije B	01111111	127	7F

Pre početka upotrebe PIO treba prvo u kontrolne registre upisati odgovarajuće podatke koji će odrediti željene načine rada (na adresu 95 za kapiju A, a na 127 za kapiju B).

Izabrani način reda	Podatak koji treba upisati Binerni	Decimalni	Heks
izlaz (način 0)	00111111	63	3F
ulaz (način 1)	01111111	127	7F
dvosmerni (način 2)	10111111	191	BF
kontrolni (način 3)	11111111	255	FF

Mogu se upisati i drugi brojevi, oni koji se od navedenih razlikuju samo u bitovima D4 i D5, pošto oni nemaju uticaja. Na primer umesto 63 može se upisati 15 (00001111).

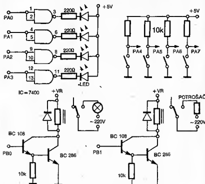
Ako je izabran kontrolni način rada, treba odmah posle 11111111 upisati u isti kontrolni registar sledeći podatek, tako de u njemu budu nule na mestima koja odgovaraju linijama koje će biti izlazi, a jedinica na onima koje odgovaraju ulazima.

Primer: ako je upisan sledeći bajt 00110001 (49 dec.) linije PX0, PX4 i PX5 će biti ulazi, a linije PX1, PX2, PX3, PX6 i PX7 izlazi.

PRIMER UPOTREBE PIO

U ovom primeru je opisano na koji način se na PIO mogu priključiti svetleće diode (LED), prekidači i relei. LED diode se uključuju preko kola 7400 zato što izlazi PIO ne mogu da obezbede dovoljnu struju.

Otpornici od 220 oma ograničavaju struju dioda. Logička jedinica na izlazima od PA0 do PA3 dovodi do pojavljivanja logičke nule na izlazu odgovarajućeg TTL kola, pa će dioda tako zasvetleti. Kada je na izlazu PIO logička nula, odgovarajuća dioda ne svetli.



Slika 7–6 Primer upotrebe Z80 PIO

Izvodi PA4 do PA7 su upotrebljeni kao ulazi za praćenje stanja na prekidačima. Kada je prekidač isključen, napon na odgovarajućem ulazu je +5V (logička jedinica) zbog postojanja otpornika od 10K. Zatvaranjem prekidača napon na odgovarajućem ulazu postaje 0 volti (logička nula).

Relei su priključeni na kapiju B, jer je samo ona sposobna da obezbedi dovoljnu struju za pobudu darlington spoja tranzistora. Mogu se upotrebiti i drugi tipovi tranzistora, zavisno od potrebe i mogućnosti nabavke. Otpornik od 10K je dodatno osiguranje da

tranzistor na provodi kada je na izlazu PIO logička nula. Dioda otklanja negativne napone koji mogu da unište tranzistor ukoliko se jave na pobudnom namotaju ralea. +Vr je napon dovoljan za pobudu ralea. Može biti 5V, 9V, 12V i td. zavisno od toga koji tip ralea može da se nabavi. U Spektrumu postoje prva dva (12V ne može da obezbedi dovoljnu struju, trebalo bi dodatno napajanje), pa je zgodno nabaviti odgovarajući tip relea. Pomoću ralea se ostvaruje električna izolacija, pa omogućavaju komandovanje uređajima koji rada sa visokim naponima i imaju veliku potrošnju.

Sada će biti dati jednostavni programi koji prikazuju način korišćenja opisanih kola.

```
10 OUT 95,BIN 00111111: REM A
je izlaz
 100 FOR N=0 TO 3: REM pocetak p
etlje
 110 OUT 31,2↑N: REM N izlaz na
1 ostali na 0
 120 PAUSE 10: REM pauza 0.2s
 130 NEXT N: REM sledeci izlaz
 140 GO TO 100: REM ponovi
```

Pri izvršavanju ovog programa, radom ća se paliti i gasiti LED diode (ako su priključene onako kako je to opisano).

U liniji 10 se određuje da kapija A radi kao izlaz (način 0). Ostali deo programa ja **FOR** petlja koja se nalazi u beskonačnoj patlji. Kada sa vradnosti indeksa petlje N manjaju od 0 do 3, redom se na izvode kapije A šalju bajti čija je vrednost 2 ↑ N, kod kojih je samo bit N na jedinici. Tako jedan izlaz prelazi na +5V, a ostali na OV i zasvetli odgovarajuća dioda. Naredba PAUSE 10 unosi kašnjenje od 0,2s.

Sleca... primerom se čita stanja na prekidačima i u zavisnosti od toga ...avlja LED diodama i releima.

```
 10 OUT 95,255: REM A kontrolni
nacin"
 20 OUT 95,BIN 11110000: REM A
ima 4 ulaza i 4 izlaza"
 30 OUT 127,BIN 00111111: REM B
je izlaz"
100 LET A=INT (IN 31/16): REM u
```

```
čitava se stanje na prekidačima"
 110 OUT 31,A: REM salje se poda
tak na A
 120 OUT 63,A: REM salje se poda
tak na B
 130 GO TO 100: REM ponovo
```

Prve tri linije služa za inicijalizaciju PIO. Kapija A ima 4 ulaza i 4 izlaza, a kapija B je samo izlazna. U liniji 100 se učitava bajt sa kapije A i pomera četiri mesta udesno. Tako se u četiri niža bita nalazi informacija o stanju na prekidačima, koja se šalje na LED diode i relea (2 najniža bita) i tako se njima upravlja.

7–3 RS 232 C interfejs

RS 232 C je jedan od standarda kojima se određuje način prenosa podataka. Koristi se pri komunikaciji računara sa drugim računarom, terminalom, štampačem i td. Radi se o dvosmernom serijskom pranosu. Postoji jadna linija za slanja i jedna linija za prijem podataka. Standard opisuje i kontrolne linija preko kojih uređaji koji komuniciraju mogu da obaveštavaju jedan drugog o stanju u kome se nalaze, ili da traže neku aktivnost

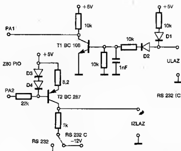
Serijski prenos znači da se šalju ili primaju bit po bit, u ovom slučaju u grupama od po 6 do 12 bita. Prvo se šalje tzv. start bit kojim se najavljuje početak prenosa, koji je uvek nula, zatim sledi 5 do 8 bita podataka. Posle njih može, ali ne mora da dođe bit parnosti, koji služi za otkrivanje grešaka pri prenosu. Na kraju sa šalju jedan ili dva stop bita koji su uvek jedinica, čime se javlja da je prenos te grupe završan.

Nivou logičke nule odgovara napon od +3 do +15 volti, a nivou logičke jedinice napon od −3 do −15 volti, a u nekim slučajevima i obrnuto. Postoji i varijanta označena sa RS 232, bez C, kod koje nivou logičke jedinice (nula) odgovara 0 volti

Brzine prenosa se kreću obično od 50 do 38000 Bd. Jedinica je bod (Bd) koji je jednak bit/sekund

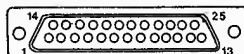
Ovaj interfejs koristi takozvani D konaktor (slika 7–7).

Sada ćamo opisati kako se može napraviti jadnostevni RS 232 (C) interfejs pomoću Z80PIO, koja je na Spektrum priključena na već opisan način



IZVOD SIGNAL
2 POSLATI PODACI (TRANSMITTED DATA)
3 PRIMLJENI PODACI (RECEIVED DATA)
7 UZEMLJENJE (GROUND)
20 PODACI SPREMNI (DATA READY)
(u ovom slučaju se ne koristi)

RS232
KONEKTOR



Slika 7–7 RS 232 interfejs

Pomoću jednog NPN tranzistora i još nekoliko elemenata naponski nivo na ulaznoj liniji se transformiše u naponske nivoe prihvetljive za PIO. Ako je na ulazu +3V do +15V, na kolektoru tranzistore će naponi biti približno OV jer će on tada provoditi, a kada je na ulazu od −3V do −15V, ne kolektoru će biti +5V jer je tranzistor tada zakočen. Dioda D1 sprečava da suviše visoki naponi ne ulazu utiču na napajanje od +5V, a dioda D2 da negativni naponi ne oštete tranzistor T1. Stanje na ulazu se preti preko PA1 izvoda PIO koji je povezan se kolektorom T1.

Podaci se na izlaz šalju preko izvoda PA2 koji komanduje tranzistorom T2. Ovaj PNP tranzistor omogućava dobijenje potrebnih naponskih nivoa koji odgovaraju standerdu. Kada je na PA2 logička nula, T2 provodi, pa je ne izlazu napon od oko −5V što odgovara ni-

vou logičke nule, a kada je na PA2 logička jedinica, T2 je zakočen pe je na izlazu napon OV ili –12V koji odgovara nivou logičke jedinica. Koji ćemo od ova dva napona (OV ili –12V) odabrati zavisi od uređaja sa kojim komuniciramo. Nivo biramo priključivanjem otpornika od 1K na odgovarejući napon. Dioda D3 i D4 i otpornik od 8,2 oma sprečavaju suviše velika izlazne struje tranzistora T2.

Opisaćemo dva potprograma koji mogu da se koriste uz ovo kolo. Prvi (SEND) šalje podatak koji se nalazi u memorijskoj lokaciji CHAR (adrasa 40959), a drugi (REC) prima podatak i smešta ga u lokaciju CHAR. Da bi se ova dva potprograma koristila treba napisati glavni program za komunikaciju koji ih poziva, a to se ostevlja korisniku jer zavisi od primene. Taj glavni program može biti napisan na mašinskom jeziku (ili, zgodnije, pomoću esemblera) ili u bejziku. Bejzik dolazi u obzir samo za menje brzine. Glevni program pri slanju podataka treba da stavi bajt koji se šalja u CHAR, zatim pozove SEND (CALL SEND za mašinski program, a RANDOMIZE USR za bejzik program), pa da stavi sledeći podatak u CHAR i pozove SEND i teko radom dok ne pošalje sve podatke. Pri prijemu treba da pozova REC (sa CALL REC odnosno RANDOMIZE USR), po povratku iz potprograma da očita dobijeni podatak iz CHAR i da ponavlja postupak sve dok ne učita sve podatke. Potprogrami su podešeni za brzinu prenosa od 1200 Bd, ali to se može izmaniti promenom konstante u potprogramu za kašnjenje DELAY.

Ovi potprogremi su nepisani asemblerskim jazikom uz pomoć programa GENS 3M firme HISOFT

```
        ORG    #A000
        ENT    #A000
CHAR    EQU    40959   ;lokacija za smest
 anje karaktera
;INICIJALIZACIJA
        LD     A,#0F   ;inicijalizacija
 PIO,B je izlaz
        OUT    (#7F),A
        LD     A,#CF   ;A je ulaz-izlaz
 nacin 3
        OUT    (#5F),A
        LD     A,#03   ;linije 0 i 1 su
 ulazi
        OUT    (#5F),A
```

```
;POTPRDGRAM ZA SLANJE KARAKTERA PREKD
RS 232
SEND    DI
        LD      BC,#001F
        LD      DE,#040B
        OUT     (C),B   ;slanje start bita
        CALL    OELAY
        LD      A,(CHAR)
LAB     RRA
        JR      C,EXT
        DUT     (C),B
OEL     CALL    OELAY
        DEC     E
        JR      Z,OP    ;da li je poslato
 svih B bita
        JR      LAB
EXT     OUT     (C),D
        JR      DEL
OP      OUT     (C),0   :slanje stop bita
        CALL    DELAY
        OUT     (C),B
        EI
        RET
DELAY   LD      HL,394 ;odredjivanje brzi
 ne prenosa
L1      NOP
        DEC     L
        JR      NZ,L1
        DEC     H
        JR      NZ,L1
        RET
;PDTPROGRAM ZA PRIJEM KARAKTERA PREKO
RS 232
REC     DI
        LO      BC,#081F
        LD      OE,0
        IN      D,(C)
```

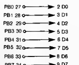
```
        BIT   1,D      :da li je primljen
  start bit
        JR    Z,NEXT-3
        EI
        RET
        CALL  DELAY
NEXT    IN    D,(C)
        SRL   D
        RRA
        CALL  DELAY
        DEC   B
        JR    NZ,NEXT
        LD    (CHAR),A
        EI
        RET
```

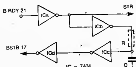
7–4 CENTRONIKS INTERFEJS

Centroniks je paralelni osmobitni interfejs koji služi za povezivanje perifernih uređaja sa računarom. Vrlo se često koristi za slanje podataka na štampač. Njegove linije su: osam linija podataka (D0–D7), $\overline{STR}$ preko koje se šalje impuls kojim sa podaci upisuju u periferni uređaj, BUSY kojom periferni uređaj logičkom jedinicom obaveštava da je zauzet i nije spreman za prijem novih podataka i GND – uzemljenje. Postoji i linija ACK (acknowladge-potvrda) koja u sledećem primeru nije upotrebljena, a kojom periferni uređaj potvrđuje prijem podataka.

Slanja podataka teče tako što računar prvo pita da li je periferni uređaj zauzet, tj. da li je BUSY linija na logičkoj jedinici. Ako jeste, ponovo pita i tako sve dok se BUSY ne vrati na logičku nulu. Tada postavlja podatak na linija podataka D0–D7 i šalje impuls (logička nula) preko linija $\overline{STR}$. Zatim, ako ima još podataka za slanje, ponovo čeka da se BUSY vrati na logičku nulu, pa zatim šalje sledeći bajt i tako redom dok ne pošalje sva podatke.

Sada ćemo opisati na koji način možemo povezati Spektrum sa štampačem ili nekim drugim uređajem posradstvom centroniks interfejsa. Iskoristićemo Z80 PIO koja je sa Spektrumom povezana na već opisani način. Podaci se šalju preko kapije B na linije podataka. Izvod PA0 kapije A je ulazni i preko njega se prati stanje na liniji BUSY. Impuls (logička nula) na $\overline{STR}$ generiše sama PIO uz pomoć





SIGNAL	IZVOD	
STR	1	(u ovom sluča-
D0	2	ju se ne koristi)
D1	3	
D2	4	
D3	5	
D4	6	
D5	7	
D6	8	
D7	9	
ACK	10	
BUSY	11	
GND	14	



Slika 7–8 Centroniks interfejs

četiri invertora. Kada mikroprocesor upiše podatak u kapiju B, izvod BRDY će preći na nivo logičke jadinice, a linija STR, zbog postojanja invertora, na nivo logičke nule. Ova promana će prako lanca invartora sa malim zakašnjenjem stići na ulaz B STB kao pojava logička jedinice, usled čega će sa B RDY vratiti na logičku nulu. Zbog toga se STR vraća na nivo logičke jedinica i tako je na toj liniji formiran impuls. Širina impulsa zavisi od kašnjenja kroz lanac invertora. Može sa proširiti dodavanjem otpornika i kondenzatora, koji su na šami označeni isprekidanim linijama.

Podaci se šalju na štampač obično kodovani ASCII kodom (Amarican Standard Coda for Information Interchange - američki standardni kôd za razmenu informacija). U njemu je svaki znek ili komanda kodovan sa 7 ili 8 bita.

Sada čemo dati jadan mali potprogram koji je napisan na asemblerskom jeziku kao i oni u primeru za RS 232 C interfejs. Da bi on mogao da radi, potrebna je inicijalizacija opisana takode u prethodnom primeru. Glavni program koji bi slao podatka treba da žaljeni podatak stavi u mamorijsku lokaciju 40959 (CHAR), zatim pozove potprogram CENT (pomoću CALL CENT ako je program na asemblerskom jeziku ili pomoću RANDOMIZE USR adresa CENT,

ako je napisana u bejziku). Ovaj postupak sa ponavlja sve dok se ne pošalju svi podaci.

```
CENT   IN    A,(#1F)  ;ucitava se stanj
e na BUSY
       BIT   0,A      ;testira se stanj
e na BUSY
       JR    NZ,CENT  ;ako je BUSY=1 po
navlja se test
       LD    A,(CHAR)
       OUT   (#3F),A  ;salje se podatak
       RET            ;povratak iz potprograma
```

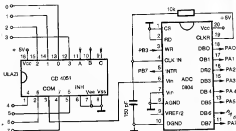
7-5 A/D KONVERTOR SA MULTIPLEKSEROM

A/D konvertor je kolo koje analogni naponski nivo pretvara u binarni broj. Na taj način se dobija digitalni podatak koji računar može da učita i zatim da ga prikaže ili da ga upotrebi za izračunavanje.

Broj bita dobijenog binarnog broja određuje koliko različitih nivoa ulaznog analognog napona konvertor može da razlikuje. Osmobitni A/D konvertor razlikuje 256 različitih nivoa (2^8), dok na primer 16 bitni razlikuje 16384 (2^{16}) različitih nivoa.

ADC 0804 ja osmobitni A/D konvertor firme National Samiconductor, izveden u samo jednom integrisanom kolu sa 20 izvoda. ADC 0803, ADC 0802 i ADC 0801 su potpuno isti u svemu, sem što su precizniji. Naponski nivo između izvoda +Vin i −Vin, koji može da se kreće od 0 do +5V, se pretvara u binarni broj od 00000000 do 11111111 (od 0 do 255 dec.). On se pojavljuje na izlazima od DB0 do DB7. Konverzija se startuje spuštanjem napona ulaza $\overline{WR}$ na logičku nulu i njegovim vraćanjem na logičku jedinicu, a završetak konverzije će konvertor označiti logičkom nulom na izlazu $\overline{INTR}$. Tada sa izvoda DB0 do DB7 možemo da učitamo rezultat i da ponovo startujemo konverziju. Vrame jedna konverzije je 100 µS.

Ovaj konvertor ima ugraden oscilator za generisanje kloka koji je potreban za rad kola. Frekvencija je određena otpornikom i kondenzatorom koji se vezuju na prikazani način. AGND je uzemljenje odnosno OV napajanja analognog dela kola, a DGND digitalnog dela. U ovom slučaju je V_{in} vezan za OV, tako da se konvertuje napon



Slika 7–9 A/D konvertor sa multiplekserom

između V_{in}^{+} i uzemljenja. Na Vref se nalazi polovina referentnog napona kola. Kada nije nigde vezan, njegova vrednost je 2,5V, pa ulazni napon može da bude od 0 do +5V. Vref se može prisilno postaviti na neku drugu vrednost i na taj način se menja opseg ulaznog napona. Ako ga postavimo na 1V, ulazni napon će moći da se kreće od 0 do 2V. Ako istovremeno vežemo V_{in}^{-} za 0,5V, opseg ulaznog napona će biti od 0,5V do 2,5V, za 0,5V na ulazu se dobija ***00000000*** na izlazu, a za 2,5V na ulazu, 11111111 na izlazu. Ovo kolo možemo i direktno vezivati za magistralu podataka mikroračunara jer izlazi DB0 do DB7 mogu da imaju visoku impedansu. Podaci se na njima pojavljuju kada se $\overline{RD}$ aktivira logičkom nulom. Dekoder adrese će aktivirati kolo preko izvoda $\overline{CS}$. U ovom slučaju kolo je uvek aktivirano i podaci su uvek dostupni jer su i $\overline{CS}$ i $\overline{RD}$ na logičkoj nuli.

Analogni CMOS multiplekser CD4051 omogućava vezivanje više ulaza na jedan A/D konvertor. Sastoji se od osam analognih elektronskih prekidača koji u uključenom stanju imaju otpornost od nekoliko stotina oma, dok im je otpornost vrlo velika kada su isključeni. Svaki od prekidača vezuje jedan od ulaza sa zajedničkim izlazom. U jednom trenutku može biti uključen samo jedan. Koji će to biti, određeno je stanjem na ulazima A, B i C.

C B A	IZABRANI ULAZ
0 0 0	0 .
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

Kada je izvod INH na logičkoj jedinici, isključuje su svi prekidači bez obzira na stanje na A, B i C. U ovom slučaju INH je vezan na logičku nulu.

A/D konvertor treba da ima posebno, stabilno i precizno napajanje odvojeno od napajanja računara, jer od toga veoma mnogo zavisi dobar rad kola. Može se upotrebiti opisano dodatno napajanje ili neko drugo još preciznije, na primer sa integrisanim kolom 723.

Spektrum dobija podatke sa A/D konvertora i njima upravlja preko Z80A PIO koja je sa računarom povezana na već opisani način.

Ovo je program pomoću koga se na ekranu prikazuju naponi (od 0 do +5V) na ulazima multipleksera.

```
  10 OUT 95,BIN 11111111: REM A
kontrolni nacin
  20 OUT 95,BIN 11111111: REM A
ulazno
  30 OUT 127,BIN 11111111: REM B
 kontrolni nacin
  40 OUT 127,BIN 11110000: REM B
 ima 4 ulaza i 4 izlaza
 100 FOR n=0 TO 7: REM ispisivan
je broja ulaza na ekranu
 110 PRINT AT n+2,0;"ULAZ ";n;"=
"
 120 NEXT n
 200 FOR n=0 TO 7: REM pocetak p
etlje
 210 OUT 63,8+n: REM adresa na B
 220 OUT 63,n: OUT 63,8+n: REM s
tart konverzije
```

```
 230 IF IN 63<128 THEN  GO TO 23
0: REM cekanje kraja konverzije
 240 LET rez=5*IN 31/255: REM uc
itavanje rezultata
 250 PRINT AT n+2,8;rez;" V
  ": REM ispisivanje rezultata
 260 NEXT n: REM kraj petlje
 270 GO TO 200: REM ponovo
```

U prve četiri linije inicijalizuje se PIO, tako da je kapija A ulazna, a kapija B ima 4 ulaza i 4 izlaza. Zatim sa na ekranu ispisuju brojevi ulaza sa kojih se čita napon. Naponi se čitaju i prikazuju pomoću naradbi u **FOR** petlji. Prvo se postavlja adresa ulaza, startuje se konverzija preko PB3, a zatim se čeka kraj konverzije ispitujući stanje na izvodu $\overline{INTR}$. Kada je konverzija gotova, učitava se rezultat i izračunava napon koji se zatim ispisuje na akranu. Postupak se ponavlja za sladeći ulaz i tako redom. Ovaj proces se ponavlja sve do prakida programa.

7–6 INTERFEJS ZA PALICE ZA IGRU

Interfajs za palice za igru se sastoji od tri LSTTL kola. Dva dekoduju adresu, a tzv. bafar podatke sa palice propušta na magistralu podataka u trenutku pojave odgovarajuće adrase. Napajanja od +5V sa uzima iz Spektruma, a požaljno je uz svako intagrisano kolo staviti kondenzator radi smanjivanja smetnji u napajanju.

Sama palica za igru se sastoji od četiri prekidača za određivanje smera i jednog koji je komanda za paljbu. Kada se palica nakrene u nekom smaru, zatvaraju se kontakti odgovarajućeg prekidača, a u međupoložajima sa zatvaraju kontakti dva susedna prekidača.

Adresa ove periferne jedinice je 31, pa podatke možemo učitati komandom **IN 31**. U trenutku izvršenja te komande aktivna su linije $\overline{\quad}$ i $\overline{IORQ}$ (logička nula), a na adrasnim linijama od A7 do A0 su redom 00011111 (31 decimalno). Tada izlazi dvoulaznih NILI kola (74LS02) IC1a, IC1b i IC1c prelaze na nivo logička jedinice, pa su svi ulazi osmoulaznog NI kola (74LS30) na logičkoj jedinici. To znači da će njegov izlaz preći na nivo logičke nule i aktivirati ulaz $\overline{E1}$, a prako IC1d koje radi kao invartor i ulaz E2 (logičkom jadinicom). Izlazi IC3, koji su do tada imali visoku impedansu, propustiće podatke sa palica na magistralu podataka. Ako je prekidač otvoren, odgovarajući

Slike 7–10 Interfejs za palice za igru



KONEKTOR

IZVOD	SIGNAL
2	DESNO
3	LEVO
4	DOLE
5	GORE
7	ZAJEDNIČKI
9	VATRA

podatak će biti logička nula zbog postojanje otpornike od 1K, a kada se prekideč zatvori, to će biti označeno logičkom jedinicom na odgovarajućoj liniji podataka. Linije podetaka D5, D6 i D7 nisu iskorišćene i uvek su ne logičkoj nuli.

Umesto kola 74LS241 može se upotrebiti kolo 74LS244, samo što tada i izvod 1 ($\overline{E1}$) i izvod 19 ($\overline{E2}$) trebe vezati na izlaz IC2.

Ovakav interfejs odgovara sistemu firma Kempston. Postoje i drugi sistemi, koji se razlikuju po tome što je rasporad prekidače na linijama podatake drugačiji i što je zajednički vod OV, a ne +5V kao ovde.

Sama palica se može kupiti i priključiti ne interfejs, ili se moža napraviti. Respored izvoda konektora ze spajanje palice sa interfejsom je det na slici.

DODATAK:

Tabela kodova

decimalno	heks.	karakter
0	0	neupotrebljeno
1	01	
2	02	
3	03	
4	04	
5	05	
6	06	PRINT zarez
7	07	EDIT
8	08	pokazivač levo
9	09	pokazaivač desno
10	0A	pokazivač dole
11	0B	pokazivač gore
12	0C	DELETE
13	0D	ENTER
14	0E	broj
15	0F	neupotrebljeno
16	10	kontrola INK
17	11	kontrola PAPER
18	12	kontrola FLASH
19	13	kontrola BRIGHT
20	14	kontrola INVERSE
21	15	kontrola OVER
22	16	kontrola AT
23	17	kontrola TAB
24	18	neupotrebljeno
25	19	
26	1A	
27	1B	
28	1C	
29	1D	
30	1E	
31	1F	
32	20	prazno polje
33	21	!
34	22	"
35	23	≠
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	–
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:

59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	/
93	5D	]
94	5E	↑
95	5F	—
96	60	£
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	8E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	©
128	80	□
129	81	▝
130	82	▘
131	83	▀
132	84	▗
133	85	▐
134	86	▚
135	87	▜
136	88	▖

137	89	▞
138	8A	▌
139	8B	▛
140	8C	▄
141	8D	▟
142	8E	▙
143	8F	█
144	90	(a)
145	91	(b)
146	92	(c)
147	93	(d)
148	94	(e)
149	95	(f)
150	96	(g)
151	97	(h)
152	98	(i)
153	99	(j) grafika
154	9A	(k) definisana
155	9B	(l) korisnikom
156	9C	(m)
157	9D	(n)
158	9E	(o)
15g	9F	(p)
160	A0	(q)
161	A1	(r)
162	A2	(s)
163	A3	(t)
164	A4	(u)
165	A5	RND
166	A6	INKEY$
167	A7	PI
168	A8	FN
169	A9	POINT
170	AA	SCREEN$
171	AB	ATTR
172	AC	AT
173	AD	TAB
174	AE	VAL$
175	AF	CODE
176	B0	VAL
177	B1	LEN
178	B2	SIN
179	B3	COS
180	B4	TAN
181	B5	ASN
182	B8	ACS
183	B7	ATN
184	B8	LN
185	B9	EXP
186	BA	INT
187	BB	SQR
188	BC	SGN
189	BD	ABS
190	BE	PEEK
191	BF	IN
192	C0	USR
193	C1	STR$
194	C2	CHR$
195	C3	NOT
196	C4	BIN
197	C5	OR
198	C6	AND
199	C7	<=
200	C8	>=
201	C9	<>
202	CA	LINE
203	CB	THEN
204	CC	TO
205	CD	STEP
205	CE	DEF FN
207	CF	CAT
208	D0	FORMAT
209	D1	MOVE
210	D2	ERASE
211	D3	OPEN #
212	D4	CLOSE #
213	D5	MERGE
214	D6	VERIFY

215	D7	BEEP
216	D8	CIRCLE
217	D9	INK
218	DA	PAPER
219	DB	FLASH
220	DC	BRIGHT
221	DD	INVERSE
222	DE	OVER
223	DF	OUT
224	E0	LPRINT
225	E1	LLIST
226	E2	STOP
227	E3	READ
228	E4	DATA
229	E5	RESTORE
230	E6	NEW
231	E7	BORDER
232	E8	CONTINUE
233	E9	DIM
234	EA	REM
235	EB	FOR
236	EC	GO TO
237	ED	GO SUB
238	EE	INPUT
239	EF	LOAD
240	F0	LIST
241	F1	LET
242	F2	PAUSE
243	F3	NEXT
244	F4	POKE
245	F5	PRINT
246	F6	PLOT
247	F7	RUN
248	F8	SAVE
249	F9	RANDOMIZE
250	FA	IF
251	FB	CLS
252	FC	DRAW
253	FD	CLEAR
254	FE	RETURN
255	FF	COPY

Indeks

LITERATURA

1 S Vickers, ZX SPECTRUM BASIC programming, Sinclair Research Ltd., Cambridge, 1983
2 N Ardley, ZX SPECTRUM + Handbuch fur Benutzer, Sinclair Research Ltd, 1984
3 I Logan, F.O Hara, The Complete Spectrum ROM Disassembly Melbourne House Publishers Ltd, United Kingdom, 1983, ISBN 0-86161-116-2
4 I Logan, Understanding your Spectrum–Basic and Machine Code Programig, Melbourne House Publisher Ltd Lincoln, 1982
5 Tehnička dokumentacija za mikroračunar Sinclair ZX Spectrum i periferne uređaje Interface 1, Interface 2 i ZX Printer, ISKRA Commerce, 1984
6. A Dickens, Spectrum Hardware Manual, Melbourne House ISBN 0–86161–115–2
7 Z80–CPU Tecnical Manual, Zilog Co, USA
8 RCA, Linear Integrated Circuits and MOS/FET s DATABOOK SSD 240B–E, 1984
9 RCA, CMOS Integrated Circuits Databook, SSD–250C, 1984
10 National Semiconductor Corp. Linear Databook, 1980
11 Fairchild TTL Databook 1980

Ostala izdanja Mikro knjige

IBM PC Uvod u rad, DOS, BASIC
Uvodi Vas u rad na IBM računarima, u MS/PC–DOS i IBM BASIC i upoznaje vas sa mnogobrojnim temama. Iz čega se sastoji računarski sistem? Kako se instalira i startuje? Rad sa tastaturom i diskom, osnove operativnog sistema. Kako se koriste gotovi programi? Šta je MS/PC–DOS? Njegova uloga, upotreba i organizacija. Sve komande DOS–a. Šta su DOS programi? Koje se greške javljaju pri radu sa DOS–om. Sve o BASIC–u, od osnovnih pojmova do kompletnog pregleda svih naredbi BASIC–a. Veliki broj primera. U čemu je razlika između Microsoft BASIC–a (BASICA), GWBASIC–a, XBASIC–a. Kako se kompiliraju BASIC programi? *320 strana formata 17x23 cm.* ISBN 86–80003–03–4

PASCAL priručnik
Prevod knjige *PASCAL User Manual and Report* (trećeg revidiranog izdanja iz 1985. god.) autora Kathleen Jensen i Niklaus Wirth. U prvom delu knjige postupno i sistematsko izlaganje materije, praćeno brojnim primerima, omogućava brzo upoznavanje i učenje Pascala. U drugom delu knjige je kompletno prikazan programski jezik Pascal u vidu skupa referentnih definicija Pascala koje su neophodne svakom programeru. PASCAL priručnik je knjiga koja će uvek ostati potpuna i nezamenljiva literatura o programskom jeziku Pascal. *256 strana formata 16x23 cm.* ISBN 86–80003–04–2

Priručnik dBASE III plus
Knjiga o najpoznatijem programu za obradu baza podataka. Knjiga o programu dBASE III plus, firme Ashton Tate. Pomoćiće vam u svim onim primenama gde je potrebno vođenje evidencije o poslovanju, materijalu, vremenu, novcu, ljudima... Koristite i vi u vašem poslovanju savremeno dostignuće koje koristi razvijeni svet. Priručnik dBASE III plus je kompletan vodič i za programe dBASE III i

dBASE II. Autori: Blaže Brdareski, Dragan Tanskoski i Vladimir Janković. *288 strana formata 17x23 cm.* ISBN 86–80003–05–0

Commodore za sva vremena
Najkompletnija knjiga o računaru Commodore 64 na našem tržištu, a verovatno i na svetskom tržištu. Detaljno obuhvata uvod u rad, BASIC, principe programiranja, Simons BASIC, mašinsko programiranje, organizaciju memorije i ROM rutine, električne šeme računara sa objašnjenjima rada i uputstva za konstruisanje raznih interfejsa. Autori: Dragan Tanaskoski, Stevan Milinković i Vladimir Janković. *344 strane formata 16x23 cm.* ISBN 86–80003–02–6

U SVIM BOLJE SNABDEVENIM KNJIŽARAMA

ili nam pišite na adresu: **Mikro knjiga**
P.O.Box 75
11090 Rakovica–BEOGRAD

SADRŽAJ

Spektrum priručnik je daleko ispred svih drugih, najbolja knjiga za one korisnike Spektruma kod nas čiji su zahtevi veći ...

MOJ MIKRO

Spektrum priručnik je vrlo vredna knjiga, omogućuje izlazak iz perioda upotrebe računala kao igračke i savladavanje potrebnih znanja za ostvarivanje vlastitih ideja u području programiranja, obrade podataka, automatika i upravljanja ...

TREND

YU ISBN 86-80003-01-8

